

Université de Montréal

Algorithms for Classifying Recorded Music by Genre

par
James Bergstra

Département d'Informatique et de Recherche Operationelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en Informatique

Août, 2006

© James Bergstra, 2006.



CA

76

U54

2006

V-048

AVIS

L'auteur a autorisé l'Université de Montréal à reproduire et diffuser, en totalité ou en partie, par quelque moyen que ce soit et sur quelque support que ce soit, et exclusivement à des fins non lucratives d'enseignement et de recherche, des copies de ce mémoire ou de cette thèse.

L'auteur et les coauteurs le cas échéant conservent la propriété du droit d'auteur et des droits moraux qui protègent ce document. Ni la thèse ou le mémoire, ni des extraits substantiels de ce document, ne doivent être imprimés ou autrement reproduits sans l'autorisation de l'auteur.

Afin de se conformer à la Loi canadienne sur la protection des renseignements personnels, quelques formulaires secondaires, coordonnées ou signatures intégrées au texte ont pu être enlevés de ce document. Bien que cela ait pu affecter la pagination, il n'y a aucun contenu manquant.

NOTICE

The author of this thesis or dissertation has granted a nonexclusive license allowing Université de Montréal to reproduce and publish the document, in part or in whole, and in any format, solely for noncommercial educational and research purposes.

The author and co-authors if applicable retain copyright ownership and moral rights in this document. Neither the whole thesis or dissertation, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms, contact information or signatures may have been removed from the document. While this may affect the document page count, it does not represent any loss of content from the document.

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Algorithms for Classifying Recorded Music by Genre

présenté par:

James Bergstra

a été évalué par un jury composé des personnes suivantes:

Yoshua Bengio,	président-rapporteur
Douglas Eck,	directeur de recherche
Sébastien Roy,	membre du jury

Mémoire accepté le 23 novembre 2006

Résumé

Ce mémoire traite le problème de la classification automatique de signaux musicaux par genre.

Dans un premier temps, je présente une technique utilisant l'apprentissage machine pour classer des statistiques extraites sur des segments du signal sonore. Malgré le fait que cette technique a déjà été explorée, mon mémoire est le premier à investiguer l'influence de la longueur et de la quantité de ces segments sur le taux de classification. J'explore également l'importance d'avoir des segments contigus dans le temps. Les segments d'une à trois secondes apportent une meilleure performance, mais pour ce faire, ils doivent être suffisamment nombreux. Il peut même être utile d'augmenter la quantité de segments jusqu'à ce qu'ils se chevauchent. Dans les mêmes expériences, je présente une formulation alternative des descripteurs d'audio nommée Mel-frequency Cepstral Coefficient (MFCC) qui amène un taux de classification de 81 % sur un jeu de données pour lequel la meilleure performance publiée est de 71 %. Cette méthode de segmentation des chansons, ainsi que cette formulation alternative, ont pour but d'améliorer l'algorithme gagnant du concours de classification de genre de MIREX 2005, développé par Norman Casagrande et moi. Ces expériences sont un approfondissement du travail entamé par Bergstra et al. [2006a], qui décrit l'algorithme gagnant de ce concours.

Dans un deuxième temps, je présente une méthode qui utilise FreeDB, une base de données d'information sur les albums, pour attribuer à un artiste une distribution de probabilité sur son genre. Avec une petite base de données, faite à la main, je montre qu'il y a une haute corrélation entre cette distribution et l'étiquette de genre traditionnel. Bien qu'il reste à démontrer que cette méthode est utile pour organiser une collection de musique, ce résultat suggère qu'on peut maintenant étiqueter de grandes bases de musique automatiquement à un faible coût, et par conséquent de poursuivre plus facilement la recherche en classification à grande échelle. Ce travail sera publié comme Bergstra et al. [2006b] à ISMIR 2006.

Keywords: classification de musique par genre, extraction de caractéristiques sonores, recherche d'information musicale, apprentissage statistique

Abstract

This thesis addresses the problem of how to classify music by genre automatically. First, I present a technique for labelling songs that uses machine learning to classify summary statistics of audio features from multiple audio segments. Though this technique is not new, this work is the first investigation of the effect of segment length and number on classification accuracy. I also investigate whether it is important that the segments be contiguous. I find that one- and three-second contiguous segments perform best as long as there are sufficiently many segments, and it helps to use more segments than are necessary to cover the song by letting them overlap. In the same experiments, I present an alternative formulation of the popular Mel-frequency Cepstral Coefficient (MFCC) audio descriptors that leads to 81% classification accuracy on a public dataset for which the highest published score is 71%. This segmentation method and new audio feature are both improvements on an algorithm by Norman Casagrande that won the MIREX 2005 genre classification contest. These experiments follow directions for future work proposed in Bergstra et al. [2006a], which describes the contest-winning algorithm.

Second, I present a technique for distilling artist-wise genre descriptors from an online CD database called FreeDB. Using a small hand-made dataset, I show that these descriptors are highly correlated with traditional genre labels. While it remains to be seen whether these descriptors are suitable for organizing music collections, this result raises the possibility of automatically labelling large datasets at low cost, thereby spurring research in large-scale genre classification. This work will be published as Bergstra et al. [2006b] in the proceedings of ISMIR 2006.

Keywords: music genre classification, audio feature extraction, music information retrieval, machine learning

Contents

1	Introduction	13
1.1	Contribution	13
1.2	Layout	14
2	Background	15
2.1	Music Genre	15
2.1.1	Traditional Genre	16
2.1.2	Collaborative Genre	17
2.2	Sound	18
2.2.1	Physics and Perception of Sound	18
2.2.2	Signal Intensity	18
2.2.3	Decibels, Phons	19
2.2.4	Loudness	19
2.2.5	Critical Bands, Mel Scale	19
2.2.6	Discrete Fourier Transform	19
2.2.7	Cepstral Analysis	20
2.2.8	Summary	21
2.3	Machine Learning of Multiclass Classifiers	21
2.3.1	Model Evaluation, Selection	22
2.3.2	Neural Networks	23
2.3.3	Other Classifiers	25
3	Genre Classification Datasets	27
3.1	Tzanetakis	27
3.2	AllMusic	28
3.3	USPOP	29
3.4	MIREX 2005	29
3.5	FreeDB	30
3.5.1	Tags per Track	31
3.5.2	Overall Tag Popularity	31
3.5.3	First-Tag Importance	33

3.6	LastFM	34
4	Genre Classification of Recorded Audio	35
4.1	Frame-level Features to Represent Music	36
4.1.1	FFTC	36
4.1.2	RCEPS	37
4.1.3	MFCC	38
4.1.4	ZCR	39
4.1.5	Rolloff	39
4.1.6	Spectral Centroid, Spectral Spread	40
4.1.7	Autoregression (LPC, LPCE)	40
4.2	Song Classification Algorithms	41
4.2.1	No Aggregation	41
4.2.2	Feature Sample Statistics	41
4.2.3	Song Segmentation	41
4.2.4	Mixture Modelling	41
4.2.5	Rhythm, Temporal Dynamics	42
4.3	Contiguous and Non-Contiguous Song Segmentation	42
4.4	Mel-Scale Phon and Sone Coefficients	43
4.5	Neural Network Classifier	44
4.6	Feature and Segmentation Performance	44
4.6.1	Mel-scale Cepstral Coefficients	46
4.6.2	MIREX.MV	47
4.6.3	Mel-scale Phon Coefficients	48
4.6.4	Mel-scale Sone Coefficients	48
4.6.5	Analysis	49
4.7	Discussion	51
5	Genre Classification of FreeDB Histograms	52
5.1	Input: FreeDB	52
5.2	Target: AllMusic	53
5.3	Predictive Model	54
5.4	Prediction Performance	54
5.4.1	Classification Rates ($1 - L0 - 1$)	54
5.4.2	KL Divergence	55
5.5	Discussion	56
6	Future Work and Conclusion	57
6.1	Coarse Genre Classification	57
6.2	High-Resolution Genre Classification	57
6.3	Conclusion	58

Bibliography	60
A Software	64
A.1 fextract	64
A.2 liblayerfb	64
B Last.FM Tag Sample	66
C Preprint: Aggregate Features and AdaBoost for Music Classification	67

List of Tables

3.1	Tzanetakis summary	27
3.2	AllMusic Genre Hierarchy	28
3.3	USPOP summary	29
3.4	MIREX 2005 dataset summary	30
3.5	MIREX 2005 Genre contest results	30
4.1	Feature summary: MIREX.MV	46
5.1	Predictive power of FreeDB genre histograms.	54

List of Figures

3.1	Histogram of Genres-per-Track for FreeDB	32
3.2	Histogram of Tag Frequencies	32
3.3	Histogram of First-Tag Importance on FreeDB	33
4.1	Pseudocode: Mel-Scale Warp	38
4.2	Pseudocode: Zero-Crossing Count without FPU	39
4.3	Contiguous Segmentation	42
4.4	Pseudocode: Delta-bar-Delta	45
4.5	The performance of the MFCC.MCV features.	47
4.6	The performance of the MIREX.MV features.	48
4.7	The performance of the MPC.MCV features.	49
4.8	The performance of the MSC.MCV features.	50
5.1	Examples of FreeDB-genre	53
5.2	Ranking of AllMusic-genre	56

List of Abbreviations

DFT	Discrete Fourier Transform
MFCC	Mel-frequency Cepstral Coefficients
MIREX	Music Information Retrieval Evaluation Exchange
PCM	Pulse Code Modulation
RCEPS	Real Cepstral Coefficients

c	a class (in context of classification)
C	a set of classes
x	a feature vector
X	a feature vector space
$\text{dct}()$	Discrete Cosine Transform
\mathcal{F}_s	Discrete Fourier transform of signal s
f	A multiclass predictor
I	Intensity (of a signal)
I_{cond}	Indicator function. 1 if cond is true, 0 otherwise
k	Frequency, used as an index
KL	Kullback-Leibler divergence
L	Loudness, in Phons
L_{0-1}	0-1 Loss over a population
L_{min}	Training error
L_{val}	Validation error
L_{test}	Test error
\hat{L}_{0-1}	0-1 Loss over a finite sample
l	Learning Algorithm
p	Pressure (of a signal)
p	A discrete probability mass function
p_i	The probability assigned by p to event i
q	A discrete probability mass function
q_i	The probability assigned by p to event i
R	A Training Set
S	A dataset of (feature, class) pairs
s	Signal
\hat{s}	Sampled Signal
T	Time, used as a bound
T	Test Set
t	Time, used as an index

Acknowledgements

I would like to thank several students and professors in the GAMME and LISA labs, whose ideas and encouragement inspired me: Norman Casagrande, Alexandre Lacoste, Dumitru Erhan, Balazs Kegl, and Yoshua Bengio.

I would like to thank Douglas Eck for his pragmatic guidance, and also for affording me the freedom to experiment. Many of the things I tried did not make it into this document, but some of them did, and it kept the Masters fun.

I would like to thank Olga on a personal level for her patience, impatience, and support, and on a technical level for her careful proofreading and numerous suggestions for the text.

I would like to thank my family for their support and understanding (especially on Thanksgiving 2005).

I would like to thank Benoit for renting us his beautiful appartement on Rue Jean Maridor this past July.

Chapter 1

Introduction

This Masters thesis addresses the problem of how to classify recorded music by genre automatically. Provisionally, consider that a genre is a set of similar-sounding musical works. Genre descriptors constitute an important navigational mechanism for music collections. The right genre descriptors can help individuals browse their own collections, but more importantly they can help listeners to search efficiently through the enormous quantity of music that is currently available (online, for example) without listening to it. As automatic collaborative and content-based indexing tools improve, individuals are empowered to circumvent the record publishing industry and connect directly with the artists they support. Not only can such tools help music fans find better music, they can help budding artists find an audience.

1.1 Contribution

This Masters thesis contributes to the project of automatic genre labelling in two ways. First, it presents a new song classification algorithm that is better than previously published ones on a public dataset. The advances are in the areas of feature extraction and of song segmentation (explained in chapter 4). A variation of this algorithm was the best genre classification algorithm at MIREX 2005, an annual music information retrieval competition conducted in conjunction with the ISMIR conference. Several variations of the technique are evaluated in chapter 4.

Second, this thesis contributes a technique for using a previously unused online repository of CD meta-data as a resource for automatically labelling artists with a genre descriptor. This genre descriptor is a broad multinomial probability distribution, but chapter 5 shows that there is a strong correlation between this notion of genre and a more traditional one from AMG (<http://www.allmusic.com>). It remains to be seen whether this notion of genre can identify similar tastes of music fans, but experiments with AMG's data show that it is a reasonable surrogate for true genre labels in the context of developing and evaluating automatic genre-classification algorithms.

1.2 Layout

This document is arranged as follows.

Chapter 2 covers required background material including discussion of what music genre is, a basic primer in the physics, perception, and processing of sound, and a brief introduction to the principles and methods of machine learning.

Chapter 3 describes datasets that represent music genre learning problems.

Chapter 4 is the first chapter of original work, in which I present and evaluate some algorithms for classifying recorded music. It should be viewed as a follow-up to Bergstra et al. [2006a], which is attached as appendix C. This chapter includes a lengthy discussion of acoustic features for classification and details of their implementations.

Chapter 5 is the second chapter of original work, in which I evaluate a collaborative online CD catalog as an alternative to a traditional (but expensive) online music reference. This chapter borrows heavily from Bergstra et al. [2006b].

Chapter 6 presents some directions for future work, and conclusions to be drawn from chapters 4 and 5.

Appendix A describes some of the more interesting software written in the course of my Masters: a library for extracting audio features, and a library for implementing neural networks. Appendix B lists a random sample of tags from the **LastFM** dataset.

Chapter 2

Background

In this section I introduce the topics of music genre, physics and psycho-acoustics of sound, the Fourier Transform, and machine learning. The section on music genre will prepare the reader for chapter 3, in which I present several types of genre datasets. The section on sound will provide some theoretical basis for the audio feature-extraction algorithms presented in chapter 4. The section on machine learning (with a focus on neural networks) provides some background on the methodology applied in chapters 4 and 5, as well as previous work in genre classification.

2.1 Music Genre

“Genres are tools used to commodify and commercialize an artist’s complex personal vision.”

-John Zorn

There are many perspectives on what music genre is, and how genre classification should be approached. For example, Pachet and Cazaly [2000] introduced a genre hierarchy that uses instrumentation and historical descriptors. Other researchers such as McKay and Fujinaga [2005] have defined self-consistent and complete taxonomies for the purposes of classifying music without ambiguity. However, these can be seen as alternatives to a mainstream view of genre that is embodied in terms like *rock*, *blues*, and *pop*. In this section, I present two views of genre that posit music genre descriptors as mechanisms for music recommendation, rather than detailed or consistent classification. What I will refer to as *traditional genre* is the one developed by the record publishing industry primarily for product placement. What I will refer to as *collaborative genre* arises from online databases in which a large number of listeners apply their own labels.

2.1.1 Traditional Genre

AMG's AllMusic Guide, a large and influential reference for music genre, arranges genre labels in a simple two-level hierarchy.¹ At the top level, they have *classical* and *popular*, beneath which we find genres like *ballet*, *concerto*, *symphony* and *blues*, *rap*, *latin*, respectively. AllMusic is described in more detail in section 3.2, but it should be understood that these genres are very broad both in terms of style and appeal. To provide more descriptive power within the restrictive hierarchy, AllMusic has introduced “Style” and “Mood” descriptors to the songs, albums, and artists in their database. The genre list that was part of the ID3 standard is another well-known list.² The original ID3 standard provided a single numerical field within each tag to record the genre to which the recording belonged. This list was similar in spirit to AllMusic's taxonomy, though the specific choice of genres was different.

Since artists do not mention genres in their creations, it is interesting to consider where genres come from. One explanation, articulated by Perrott and Gjerdingen [1999], is that the record publishing industry encouraged genres as a simple means to describe their different artists and acts. The broad genres such as *rock* and *blues* mentioned above can be seen in this light as components of a general method of music recommendation. There is no reason to suppose a priori that such genres serve as a taxonomy for music; indeed there is considerable evidence that popular genre sets make poor taxonomies. Pachet and Cazaly [2000] outline a number of problems associated with using musical genres common in the music industry; Aucouturier and Pachet [2003] summarize:³

1. They are designed for albums, not tracks.
2. There is considerable disagreement among different taxonomies on how to classify individual albums.
3. Within these taxonomies, taxons do not bear fixed semantics, leading to ambiguity and redundancy. (eg. Genres *jazz* and *christmas* overlap.)
4. They are sometimes culture-specific, and often not related to actual musical content. (eg. A genre such as *christian*.)

Despite theoretical reasons for why genres are not taxonomies, genre papers from ISMIR proceedings (e.g. Fingerhut [2002] - Crawford and Sandler [2005]) have universally treated the problem as one of classification. The target class sets have been of a small size, such as five to ten genres. If it seems like an artificially simple task, consider that it is at least one at which people are relatively good. If computers could represent the recordings in a suitable way, then it would be easy for computer programs too. The **USPOP** dataset (described in section 3.3) and a hand-labelled dataset using 10 genres according to George Tzanetakis (described in section 3.1) have been popular datasets for comparing algorithms. In both of these datasets, the genre groups are very coarse, and of limited value for making recommendations.

¹AllMusic is described more thoroughly in section 3.2

²ID3 is a standard for embedding meta-information in mp3 audio. For details see <http://www.id3.org>.

³The examples have been added by the authors, we hope that they are consistent with the original intent.

It should be borne in mind throughout this thesis that genre classification, on its own, is not a relevant practical problem. As long as the number of genres is small, and defined by a few human experts, it is not difficult for those experts to label new music on demand. While research in genre classification has resulted in interesting algorithms and audio representations for music classification, the final measure of these algorithms and representations must come from elsewhere.

2.1.2 Collaborative Genre

An emerging alternative to the traditional view of genre comes from Internet-based services such as FreeDB (the subject of chapter 5) and Last.FM (discussed in section 3.6. FreeDB operates as a tagging service for CDs, and Last.FM operates customized radio streams for individual clients. Both of these services collect anonymous suggestions for the genre of specific albums or tracks, and do not constrain suggestions to any sort of menu. Wherever one user attaches one genre, another user may attach another and the system retains both labels. Songs, albums, and artists accumulate histograms over the growing number of labels in the system. The result is that these services have thousands of labels, though the labels aren't organized into any hierarchy. These labels overlap, there is no attempt at taxonomy. Some labels are broad, some narrow, some labels are words that have a meaning related to the music, the artist, the region or period of recording, and many labels in Last.FM's collection have amusing meanings in English that are not necessarily related to the music at all (e.g. *pancakeday*, *wiggidywopwoo*. See appendix B for a random sample of lastfm tags.).

On one hand, we may see this histogram as quantifying uncertainty in the true genre of a particular song or artist; but on the other hand, we may see the histogram as actually being a point in a continuous vector space of genre. These are both interesting possibilities that lead to subtly different learning problems. In the former case, the idea that there is a true genre suggests that a multiclass classification algorithm would be appropriate. In the latter interpretation, a regression into multinomial parameters would be more natural.

Last.FM uses their labels as agents for recommendation, just like AllMusic, but Last.FM does not choose the genres or constrain their meaning as they evolve. Last.FM's Tag Radio service plays the set of music with a certain tag as a radio station. Playing *rock* selects a very broad set of music, but smaller genre sets such as *electropop* generate very coherent radio, similar to the playlist of a specialty radio show. The large number of labels and semantics in this distributed notion of genre poses several problems, because no single person can label a new artist, album, or song to inject it into an established collection. In a nutshell: it is impossible to hear a new song on Tag Radio, because many people must listen to it (and tag it) first. Another problem is that it can be important to balance the relative strength of a song's membership in different genres; a simple member vs. no-member decision is not natural. Furthermore, the lack of a natural hierarchy in the label set means that tracks with only a specialized label cannot be automatically included into more general labels' sets. Learning to predict genre as a collaborative distribution is a difficult and relevant learning problem that is closely related to the learning problem associated with genre

as a taxonomy, but which introduces new challenges in data-mining and computational efficiency. This topic will be revisited in chapter 5, in which I evaluate the correspondance between FreeDB and AllMusic, but many questions remain for future work (chapter 6).

2.2 Sound

This section begins with a brief survey of relevant topics in physics, psychoacoustics and signal processing, in preparation for discussion of audio feature extraction in chapter 4. This section is paraphrased from Gold and Morgan [2000], an introduction to signal processing and machine learning for speech analysis and synthesis. This background information motivates some of the standard transformations that extract features from audio for music classification, as well as the two features that will be introduced in chapter 3.

2.2.1 Physics and Perception of Sound

The basic premise of the physics of sound is that sound can be seen as a superposition of waves of different frequency, phase, and amplitude. Physically, sound is transmitted through air as oscillations in pressure with respect to time and space. Sound is transcribed to and from electric current using the same electro-mechanical device: a diaphragm attached to a magnet housed in a spool of insulated wire. In a microphone, oscillations in air pressure pull and push on the diaphragm and generate alternating current in the wire. In a loudspeaker, the reverse process uses alternating current in the wire to drive the diaphragm to create oscillations in air pressure. A digital sound signal is obtained by frequently and regularly measuring (*sampling*) the voltage on the wire from the microphone. The encoding of an audio signal by writing the values of these voltage samples is called Pulse Code Modulation (PCM). It is the encoding of sound that is used in .wav files, .au files, and CD-audio. CD audio, for example, is sampled at 44100Hz (Hertz, or Hz, denotes a number of events per second).

2.2.2 Signal Intensity

Intensity is defined as the amount of energy flowing across a unit area surface in a second. This is equivalent to the pressure p multiplied by the velocity v . The equation 2.1 shows how intensity I relates to pressure p and velocity v . Under normal conditions (of low-amplitude waves) the velocity of air molecules varies linearly with the root-mean-squared (RMS) sound pressure. In this case the intensity I is proportional to the square of the pressure p .

$$I = \bar{p}\bar{v} \propto \bar{p}^2 \quad (2.1)$$

In this document, the terms pressure and intensity will always denote their respective RMS definitions.

2.2.3 Decibels, Phons

Decibels measure the difference in energy between two signals. The definition of decibels, L , in equation 2.2 can be expressed either in terms of intensity or pressure, but in both cases a reference intensity or pressure is required.

$$L = 10 \log_{10} \left| \frac{I_1}{I_{ref}} \right| = 20 \log_{10} \left| \frac{p_1}{p_{ref}} \right| \quad (2.2)$$

This logarithmic energy scale is also used as a guide to the perceptual sense of loudness. Using a particular reference intensity I_{ref} , this is referred to as the Phon scale.

2.2.4 Loudness

Another measure of loudness is the *sone*, S . Empirical work suggests that for pure tones (sound consisting of a single frequency), the sense of loudness is consistent with equation 2.3. In equation 2.3 S varies linearly with the sone scale, p denotes pressure and I denotes intensity.

$$S \propto p^{0.6} \propto I^{0.3} \quad (2.3)$$

The constant of proportionality in equation 2.3 depends on the frequency of the wave in question, which gives rise to so-called *equal loudness curves*, that give equivalent loudnesses in terms of intensity at different frequencies. In general however, tones are not pure and the sense of loudness of a sound depends not only on their own intensity and frequency, but also the intensity and frequency and relative phase of other tones in the signal. The importance of loudness transformations in feature extraction for music information retrieval was the subject of Lidy and Rauber [2005], and many genre classification algorithms published in the last few years use some form of loudness transformation. See Gold and Morgan [2000] for more details on the perception of loudness.

2.2.5 Critical Bands, Mel Scale

Another important psycho-acoustic influence is the way pitch is perceived. The Mel-scale is a psycho-acoustic frequency scale on which a unit change carries the same perceptual significance over the entire scale. This contrasts with the simple Hertz scale. While it is easy to differentiate two tones at 440Hz and 441Hz, it is much more difficult to differentiate 4400Hz from 4401Hz. The Mel-scale increases identically with Hertz from 0 to 1000Hz, at which point it continues to rise logarithmically while maintaining a continuous derivative. The effect is that a fixed precision on the Mel-scale gives excellent resolution at low frequency, and coarse resolution of high frequencies.

2.2.6 Discrete Fourier Transform

To paraphrase the Fourier Theorem, any periodic signal can be expressed as a sum of sinusoidal components of various frequencies, each with a certain phase. The Fourier Transform is the

transform that tells us, for a given signal s , the amplitude and phase of the sinusoidal component at each frequency. Unfortunately, when working with audio, we have only a finite number of regularly-spaced samples of the true signal, so the true Fourier Transform does not apply. Instead, we defer to the Discrete Fourier Transform (DFT) given in equation 2.4, in which k is a frequency index, t is a sample (time) index running from 0 to some upper bound T , and \hat{s} is our sampled signal.

$$\mathcal{F}_s[k] = \sum_{t=0}^{T-1} \hat{s}[t] \left(\cos(2\pi \frac{kt}{T}) - i \sin(2\pi \frac{kt}{T}) \right) \quad (2.4)$$

Note that in the DFT, we only measure the amplitude and phase of frequencies indexed by k . k may run from 0 to any positive upper bound, but an interesting effect will be observed. Consider that we can use k to index a signal \hat{c}_k from equation 2.4 whose t 'th value is $c_k[t] = \cos(2\pi \frac{kt}{T})$. After k passes half the sample-length (T) of the signal then a curious thing happens to \hat{c}_k : it appears to be a sinusoid with lower and lower frequency. The culprit is the sampling frequency—the sinusoid oscillates too fast to be accurately measured—and the consequence is that $\hat{c}_{T/2+a} = \hat{c}_{T/2-a}$ for any integer a (assuming T is even). For this reason, we say that an audio signal of length T has a Nyquist position of $T/2$, or alternately that an audio signal sampled at r Hz has a Nyquist frequency of $r/2$. A digital approximation of the original signal cannot record frequencies above the Nyquist frequency, and records frequencies approaching the Nyquist with reduced fidelity. As a corollary, it can be seen that the frequency indexed by a given k is related to the Nyquist frequency of the signal. If the Nyquist frequency of the signal is r_{nyq} Hz, then the k 'th frequency is $\frac{2kr_{nyq}}{T}$ Hz.

In terms of the acoustic wave that \hat{s} represents, the result of the DFT transform is a vector of complex values, f_k , that represents the amount and phase of energy at each of the $T/2$ indexed frequencies. The energy (intensity) observed in the k 'th frequency over the length of the frame is the scalar value $|f_k|^2$. Similarly, the intensity observed in the k 'th frequency interval is $|f_k|$. Almost all acoustic features for music classification are based on the magnitudes $|f_k|$, although Bello et al. [2005] has shown that the phase component can be just as effective for detecting note onsets.

The discrete cosine transform (DCT, or in equations, `dct()`) is a name for a DFT in which all imaginary components in the output are discarded. A more formal treatment of discrete and continuous signal analysis is given in Alan W. Oppenheim [1996].

2.2.7 Cepstral Analysis

Cepstral analysis is a technique developed especially for speech recognition, for quantitatively describing the quality of voiced sound. The technique is to compute the DFT of a short segment of audio (as short as 20ms) in which a human voice is speaking; to compute the loudness (on Phon scale, for example) of each complex element of the DFT (signal energy); and to perform a second DFT on the vector of loudness values. The two DFTs play subtly different roles. The first DFT (of the recorded signal) decomposes the utterance into its spectral components so that individual frequencies can be measured, and adjusted for loudness. The second DFT may

seem strange because the basis of sinusoids has no natural semantics when applied to a vector of loudness values, as it does when applied to a signal of recorded audio. Nevertheless, the second DFT has the effect of isolating descriptors of the general shape of the loudness vector in a few dimensions corresponding to low-frequency sinusoids. Furthermore, the number (and meaning) of low-frequency descriptors is independent of the resolution of the loudness vector (given the Nyquist frequency of the original recording). In speech recognition, the general shape of the loudness vector is very helpful in predicting the shape of the mouth, and phone (unit of pronunciation) being produced. Real Cepstral coefficients (RCEPS) are the real components of the complex vector resulting from the second DFT.

One variation on the method presented above adds a linear transformation of the DFT before adjusting for loudness. For example, the MelScale Frequency Cepstral Coefficients (MFCC) are computed by averaging wider and wider frequency ranges according to the Mel scale. Numerous engineering tricks are often used to tweak this general recipe for speech recognition, and there is no one correct implementation of this method.⁴

For more information on Cepstral analysis see Gold and Morgan [2000].

2.2.8 Summary

In this section, I presented a number of topics in physics and psycho-acoustics. These topics comprise motivation and some theoretical basis for the frame-level acoustic features that will be presented in chapter 4.

2.3 Machine Learning of Multiclass Classifiers

In this section, I present some of the basics of machine learning, in the context of multiclass classification. I begin with the formal setup of the classification problem, and move on to methods of model evaluation and selection. I describe neural networks, the particular class of predictor that I'll use in my own experiments, and briefly describe some other classes of predictors such as decision trees, kernel machines, and probability models that are used in previous work. This contents of section are key to understanding the methodology of the experiments in chapters 4 and 5.

The basic premise of a multiclass classification problem is that there is some underlying phenomenon that we can model reasonably well as a joint probability distribution of examples that have both features x from some set of possible features X and classes c from a finite set of possibilities C . This distribution gives rise to the conditional probability $P(c = C|x = X)$. Machine learning of a multiclass classifier is the problem of guessing (learning) a predictor (machine) $f : X \rightarrow C$ that predicts the mode of this conditional probability distribution, when we have a sample S of examples x_i, c_i . A procedure that guesses a predictor is called a *learning algorithm*. Throughout this document, the sample S will be referred to as a *dataset*.

⁴Dan Ellis publishes Matlab code on his website that emulates MFCC calculations of several software packages.

2.3.1 Model Evaluation, Selection

Before I can define model evaluation and model selection, I must explain the important notion of a *loss function*. A loss function measures how bad a particular predictor is, either with respect to the underlying distribution or a particular set of examples. In classification problems, the *0-1 loss* (L_{0-1}) defined in equation 2.5 is a common choice. In equations 2.5 and 2.6, f denotes a predictor, \mathcal{X}, \mathcal{C} denote random variables of the feature and class, and (x_i, c_i) denote an example from a dataset T .

$$L_{0-1}(f) := P(\mathcal{C} \neq f(\mathcal{X})) \quad (2.5)$$

$$\hat{L}_{0-1}(f; T) := |T|^{-1} \sum_{(x_i, c_i) \in T} \mathbf{I}_{y_i \neq f(c_i)} \quad (2.6)$$

Model evaluation can be viewed as the process of estimating a particular population loss, such as L_{0-1} . *Model selection* can be viewed as the process of choosing the one predictor (especially from among a finite set) that realizes the smallest loss.

Independent Test Set, Cross-Validation, Bootstrap

An *independent test set* (or just *test-set*) is a subset of samples $T \subset S$ from the dataset, that follows the same distribution as the dataset, but which is not provided to the learning algorithm. The remainder of the dataset $R = S/T$ is the *training set*, which is provided to the learning algorithm. When R and T are independent, we can use R to produce a predictor f , and apply equation 2.6 to compute an unbiased estimate of $L_{0-1}(f)$ using T .

In some cases, we would prefer an evaluation of the learning algorithm itself, rather than a particular predictor. One natural measure of the performance of the learning algorithm l is the expected loss $L_{0-1,l}$ in equation 2.7, in which the elements of R are considered random variables, and $l(R)$ denotes the predictor generated by l on dataset R . A single value of $\hat{L}_{0-1}(f_R; T)$ is an unbiased estimate of our learning algorithm's performance on datasets from the given distribution of size $|R|$.

$$L_{0-1,l} = E_R[L_{0-1}(l(R))] = E_{R,T}[\hat{L}_{0-1}(l(R); T)] \quad (2.7)$$

The method of *cross-validation* is a variation on the test-set method of model evaluation that is more statistically efficient, especially for small datasets, though more computationally costly. Cross-validation involves partitioning the dataset S into some number k of disjoint subsets called *folds* (eg. 5 or 10, as suggested in Breiman and Spector [1992]). We use the folds to obtain k estimates of $L_{0-1,l}$ by choosing each fold to be T once, and using the rest of the folds as R . Since training examples are shared between estimates, the estimates are not independent. Consequently, the trial mean is an unbiased estimate of the population mean, but the trial variance is generally less than the underlying population variance (Bengio and Grandvalet [2004]). When I present K -fold mean and variance estimates for my experiments in chapters 4 and 5, the reader should bear this bias in mind.

2.3.2 Neural Networks

In this section I will explain a particular type of multiclass classifier called a neural network, that I will use in chapters 4 and 5. A neural network is a function from example features and model parameters to a distribution over class predictions, that is differentiable with respect to the parameters. Neural networks are useful for learning if we have a differentiable real-valued fitness measure (called a *cost function*) that decreases with the suitability of our network for a particular application (training set). In such a case, we can use a gradient-based optimisation method (see LeCun et al. [1998]) to find model parameters that approximately minimize the cost, and hence maximizes the suitability of the neural network function.

$$f_{net} : \mathcal{X} \times \mathcal{H} \rightarrow D(\mathcal{T}) \quad (2.8)$$

$$Cost : D(\mathcal{T}) \times D(\mathcal{T}) \rightarrow \mathbb{R} \quad (2.9)$$

The model form of a neural network is given in equation array 2.9, in which \mathcal{X} is the example feature space, \mathcal{H} is the machine parameter space, and $D(\mathcal{T})$ is the set of distributions over the target classes \mathcal{T} . Informally, f_{net} maps a feature and a particular parameter choice to a distribution over classes. It is worth noting that f_{net} is subtly different from a multilabel classifier, because it yields a distribution over classes and not a class choice. One natural way of making a classifier from f_{net} is to interpret the output distribution as $P(class|feature)$ and choose the mode. Choosing the mode cannot be part of the network itself, because the act of choosing the mode is not differentiable with respect to the distribution.

Network Structure

Often a neural network for classification comprises a composition of functions with a softmax (equation 2.10) at the highest level. Lower functions are often chosen to be linear transformations, logistic functions (equation 2.11), or radial-basis functions (described in Bishop [1996]), though any differentiable function is acceptable. My own networks for classification in chapters 4 and 5 use the softmax, logistic, and linear transforms.

$$softmax_i(x) := \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.10)$$

$$logistic_i(x) := \frac{1}{1 + e^{-x_i}} \quad (2.11)$$

$Cost$ (recall 2.9) cannot correspond exactly to \hat{L}_{0-1} , because the act of comparing the modes of two distributions p and q is not differentiable with respect to either distribution (this mode-match operator is defined in equation 2.12).

$$p \neq_m q := \begin{cases} 0 & \text{if } \arg \max_i p_i = \arg \max_i q_i \\ 1 & \text{otherwise} \end{cases} \quad (2.12)$$

$$KL(p||q) := \sum_i p_i \log \frac{p_i}{q_i} \quad (2.13)$$

Instead, for our *Cost* we must choose a differentiable function as a surrogate for the mode-match operator. Kullback-Leibler divergence, or relative entropy, is one acceptable surrogate (equation 2.13), and I will use this one in my experiments later. The Kullback-Leibler divergence is proportional to the number of bits wasted by encoding events generated according to a distribution P with a code devised to be optimal under a distribution Q . This measure is appropriate in the context of classification in the sense that $KL(p||q)$ is very large when some $p_i = 1$ while $q_i \approx 0$. It is worth noting that KL not a perfect surrogate for \neq_m . For example, with $p = [1, 0]$, \neq_m does not distinguish between $q_a = [0.99, 0.01]$ and $q_b = [0.51, 0.49]$, though $KL(p||q_a) \approx 0.004$ and $KL(p||q_b) \approx 0.29$. On the other hand, for $q_c = [0.49, 0.51]$, $KL(p||q_c) \approx 0.31$ though \neq_m jumps from 0 to 1 between q_b and q_c . See Nguyen et al. [2006] for some examples of other divergences, but no differentiable function can exactly match \neq_m so *Cost* cannot estimate it.

Learning (Optimization and Regularization)

When learning with neural networks, we ultimately wish to find parameters h that define a predictor $f_{net}(\cdot, h)$ that will realize a low loss, such as L_{test} defined below in equation 2.16. Unfortunately, the closest we can come to this quantity with a differentiable function of h is L_{min} from equation 2.14, in which h denotes a particular parameter choice and D represents a function that presents the target class c as a distribution over classes with all the probability mass at the correct class. (It is important to note that minimization considers only elements of $R_{min} \subset R$. The reason for this will be explained below.) Since *Cost* and f_{net} are differentiable with respect to h , this quantity L_{min} can be minimized with respect to h by gradient descent. For a treatment of gradient-based minimization methods, see chapter 7 of Bishop [1996] or LeCun et al. [1998]. It is worth noting that neural networks typically represent high-dimensional non-convex optimization problems that are difficult to solve, but good local minima are found in many applications.

$$L_{min}(h) := \sum_{(x_i, c_i) \in R_{min} \subset R} Cost(f_{net}(x_i, h), D(c_i)) \quad (2.14)$$

$$L_{val}(h) := \hat{L}_{0-1}(f_{net}(\cdot; h); R_{val}) \quad (2.15)$$

$$L_{test}(h) := \hat{L}_{0-1}(f_{net}(\cdot, h); T) \quad (2.16)$$

It is not clear why minimizing the quantity L_{min} on the set R_{min} should yield a predictor that minimizes $\hat{L}_{0-1}(f_{net}(\cdot, h); T)$, and indeed in general it does not. In practice, it is almost always observed that an arbitrary parameter choice h finds neither L_{min} nor L_{test} at a minimum,

and finds that an iterative minimization algorithm applied to L_{min} will also make progress when viewed as a minimization of L_{test} . After some number of iterations the minimization of L_{min} conflicts with that of L_{test} and after that happens, continued minimization of L_{min} yields worse and worse values of L_{test} . This phenomenon is called *overfitting*. Overfitting also occurs when L_{test} has the same form as L_{min} , but is calculated as a sum over different dataset examples.

One way to combat overfitting is to constrain the search-space from which h is chosen so that as L_{min} is minimized, h cannot stray into regions where \hat{L}_{0-1} will decrease. Unfortunately, for complicated network functions it is often difficult to know in advance which areas of the parameter space will be bad for a particular problem.

Another way to combat overfitting is to remove some training examples from R to make what is called a *validation set*, which I have introduced above as R_{val} . Since these examples are not used to minimize L_{min} , they can serve to estimate $L_{val} \approx L_{0-1}$, just like the examples of the test set T . The technique of *early stopping* is to iteratively minimize L_{min} until L_{val} is no longer improved by further minimization, and choose the parameters h that minimize $L_{val}(h)$ as our best guess for what will minimize $L_{test}(h)$. With respect to the technique of constraining the search space, this technique has the advantage of not requiring an understanding of the topology of the search space. One disadvantage is that the number of data examples set aside as R_{val} comes at the expense of the number of examples in R_{min} . Neural networks (as well as most learning algorithms) perform better when more training data are available, so this reduction in the cardinality of R_{min} introduces a pessimistic bias in our overarching program of model evaluation.

2.3.3 Other Classifiers

While I do not use the following learning algorithms in my own experiments, they are used throughout the previous work discussed in chapter 4.

Decision Tree

A decision tree is a binary tree whose nodes correspond to hyperplanes splitting the feature space, and whose leaves correspond to classification decisions. Equation 2.17 describes a decision tree function $f_{dtree}(x; \theta)$, in which θ is either model parameters of the form $\theta = (w, b, \theta_l, \theta_g)$, or a classification decision $\theta = c \in C$.

$$f_{dtree}(x; \theta) = \begin{cases} c \in C & \text{if } \theta = c \text{ denotes a leaf} \\ f_{dtree}(x, \theta_g) & \text{if } w \cdot x > b \\ f_{dtree}(x, \theta_l) & \text{otherwise} \end{cases} \quad (2.17)$$

Two restrictions are often used in practice. First, the form of the weight vector w is restricted to be zero everywhere except in a single dimension. This restriction justifies several fast learning algorithms based on recursive information gain. Decision Trees can easily overfit a given dataset, so pruning algorithms or other forms of regularization are necessary. For more information, see chapter 8 of Duda et al. [2001].

Kernel Machines

Kernel Machines have the general form given in equation 2.18, in which α is a vector of weights on data examples (x_i, y_i) and K is a kernel function that reports a positive similarity of two examples' features.

$$f_{kernel}(x; \alpha, K) = \sum_i \alpha_i K(x, x_i) \quad (2.18)$$

Support-vector machines (SVMs) (chapters 9,10 of Vapnik [1998]), Parzen windows (chapter 4 of Duda et al. [2001]), K-nearest neighbours (KNN) (chapter 4 of Duda et al. [2001]) are popular algorithms that correspond to particular choices of K and algorithms for determining α .

Class Probability Models

Another strategy for classification is to build a set of class-wise probability models over the feature space of the form $P(x|c)$. If we also have a table of the values $P(c)$ then we can classify new points according to Bayes rule, as in equation 2.19.

$$f_{prob}(x; P) = \arg \max_{c \in C} P(c|x) = \arg \max_{c \in C} P(x|c)P(c) \quad (2.19)$$

There are as many algorithms for performing this sort of classification as there are methods for constructing $P(x|c)$. For real-valued feature spaces, Gaussians and Gaussian mixtures are popular. For more details refer to chapter 2 of Bishop [1996] or chapter 10 of Duda et al. [2001].

Chapter 3

Genre Classification Datasets

While genre classification is infamous as a task with little theoretical basis, there are several examples of concrete learning tasks on which classification algorithms can be compared. In this section I review some of the more popular and influential datasets used to compare genre classification algorithms: **Tzanetakis**, **AllMusic**, **USPOP**. Some papers (e.g. Tzanetakis et al., Li et al. [2003]), have used in-house databases that were never released.

3.1 Tzanetakis

The **Tzanetakis** database was first used in Tzanetakis and Cook [2002], later in Li and Tzanetakis [2003], and Bergstra et al. [2006a]. This database contains 1000 audio clips, each of which is a 30-second piece of a longer commercially-produced song. The clips have been downmixed to a single audio channel, and distributed at a samplerate of 22050kHz in Sun's .au format. Each excerpt is labeled as one of ten genres (*blues*, *classical*, *country*, *disco*, *hiphop*, *jazz*, *metal*, *pop*, *reggae*, *rock*). The standard task induced by this dataset is to classify each excerpt into the correct genre.

	Songs	Genres	Audio
Tzanetakis	1000	10	yes

Table 3.1. Summary of the **Tzanetakis** dataset.

Although the artist names are not associated with the songs, my impression from listening to the music is that no artist appears twice. The so-called *producer effect*, observed in Pampalk et al. [2005a] is therefore not a concern with this dataset.

The producer effect is an important one to keep in mind when working with recorded audio. Songs from the same album tend to look overly similar through the lens of popular feature-extractors, on account of album-wide production and mixing effects. Hence, it is important when running machine learning experiments to ensure that no album is split between training and test

set. Failure to do so would bias test-set performance optimistically.

Results on this dataset are described in Tzanetakis and Cook [2002] (61%), Li and Tzanetakis [2003] (71%), Bergstra et al. [2006a] (83%). More details on these methods are given in chapter 4.

3.2 AllMusic

AllMusic¹ is a large commercial database offering a range of information about artists, albums and songs, such as “Genre”, “Style”, “Mood”, “Similar Artists” and “Followers”. AllMusic offers a genre for every artist in its comprehensive database. The most popular genre by far is *Rock*, with about half of popular artists; other popular genres include *Blues*, *Jazz*, and *Easy Listening*. Classical music is not a genre in **AllMusic**. Instead, classical music is divided among 13 genres such as *Ballet*, *Choral Music*, and *Symphony*. AllMusic’s full genre taxonomy is listed in table 3.2.

Popular		Classical
Avant-Garde	R&B	Ballet
Blues	Rap	Band Music
Cajun	Reggae	Chamber Music
Celtic	Rock	Choral Music
Comedy	Soundtrack	Concerto
Country	Vocal	Minimalist
Easy Listening	World	Film Music
Electronica		Keyboard Music
Folk		Musical Theater
Gospel		Opera
Jazz		Orchestral Music
Latin		Symphony
New Age		Vocal Music

Table 3.2. AllMusic offers a two-level hierarchy for genre. The first level distinguishes popular music from classical, the second level in each branch is given in the columns of the table.

In recognition of the fact that genres are not as precise as many listeners would like, AllMusic has introduced a number of narrower descriptors: sub-genre categories called “Style” and extra-genre descriptors called “Mood”. While only one genre per entry is allowed, multiple Style and Mood labels are often applied to a single artist, and they are not arranged in any hierarchy.

There are three factors that make AllMusic not directly usable as a dataset for music classification. The first is that AllMusic is exclusively a source of meta-information, it has no audio component. The second is that AllMusic is prohibitively expensive to licence and difficult to access automatically via the website. The third is that AllMusic is an evolving taxonomy; the Genre tags are relatively persistent (though their number was recently raised), but the Mood and

¹AllMusic is hosted online at: <http://allmusic.com>

Style descriptors are in more rapid flux. In addition, it is important to remember that AllMusic is foremost a commercial enterprise, not a music classification project for research.

3.3 USPOP

USPOP² is a database of full-length songs, together with their AllMusic meta-information. In contrast to the Tzanetakis database, it is not available as audio—Dan Ellis distributes all the music content of the database as pre-computed Mel-scale Frequency Cepstral Coefficients (described later in section 4.1.3).

The songs in **USPOP** were selected to represent popular music. The database is centred around 400 popular artists, as determined by a trawl of the OpenNap peer-to-peer network, circa 2001.³ One or more albums was purchased for each popular artist, and the tracks on those albums make up the music component of the **USPOP** database. The labels were determined by querying AllMusic.com for each one of the artists (circa 2001). Further details are available at the **USPOP** website.

It is possible to attempt several learning tasks using the **USPOP** database. Each song is associated with a single genre and a single artist. While prediction of the Style, Mood, and Similar Artists would be interesting, to my knowledge this has not been attempted.

	Entries	Albums	Artists	Styles	Genres
USPOP	8764	706	400	251	10

Table 3.3. Summary statistics of the **USPOP** database.

3.4 MIREX 2005

MIREX (Music Information Retrieval Evaluation eXchange) is an annual series of contests, starting in 2005, whose main goal is to present and compare state-of-the-art algorithms from the music information retrieval community.⁴ It is organized in parallel with the ISMIR conference (eg. Crawford and Sandler [2005]). The contest encourages many aspects of music information retrieval, embracing both symbolic and signal audio. The most popular contest was the genre competition (with 13 successful participants) that was a pair of straightforward multiclass classification tasks. The two datasets used to evaluate the submissions were based on Magnatune⁵, and **USPOP**⁶. Both databases contained mp3 files of commercially produced full-length songs. The Magnatune database had a hierarchical genre taxonomy with 10 classes at the most detailed

²www.ee.columbia.edu/~dpwe/research/musicsim/uspop2002.html

³OpenNap is an open-source server for the napster a peer-to-peer file-sharing protocol.

⁴For example, MIREX 2005 was organized primarily via a wiki at <http://www.music-ir.org/mirex2005>.

⁵Magnatune is a record label that licences music under the creative common licence. <http://www.magnatune.com>

⁶www.ee.columbia.edu/~dpwe/research/musicsim/uspop2002.html

level (*ambient, blues, classical, electronic, ethnic, folk, jazz, new age, punk, rock*), whereas **USPOP.MIREX05** was a simpler version of **USPOP** with 6 genres (*country, electronic and dance, new age, rap and hip hop, reggae, rock*). Table 3.4 summarizes the two datasets.

	Entries	Albums	Artists	Styles	Genres
USPOP.MIREX05	1515	-	77	-	6
MAGNA.MIREX05	1414	-	77	-	10

Table 3.4. Summary statistics of the databases used in MIREX 2005.

The overall performance of each entry was calculated by averaging the raw classification accuracy on **USPOP.MIREX05** with a hierarchical classification accuracy on **MAGNA.MIREX05**.⁷ Table 3.5 summarizes the contest results.⁸

Rank	Participant	Overall	Magnatune	USPOP
1	Bergstra, Casagrande & Eck[1]	82.34%	75.10%	86.92%
2	Bergstra, Casagrande & Eck[2]	81.77%	74.71%	86.29%
3	Mandel & Ellis	78.81%	67.65%	85.65%
4	West, K.	75.29%	68.43%	78.90%
5	Lidy & Rauber [1]	75.27%	67.65%	79.75%
6	Pampalk, E.	75.14%	66.47%	80.38%
7	Lidy & Rauber [2]	74.78%	67.65%	78.48%
8	Lidy & Rauber [3]	74.58%	67.25%	78.27%
9	Scaringella, N.	73.11%	66.14%	75.74%
10	Ahrendt, P.	71.55%	60.98%	78.48%
11	Burred, J.	62.63%	54.12%	66.03%
12	Soares, V.	60.98%	49.41%	66.67%
13	Tzanetakis, G.	60.72%	55.49%	63.29%

Table 3.5. Summarized results for the Genre Recognition contest at MIREX 2005. Square brackets indicate the index among multiple contest entries.

3.5 FreeDB

FreeDB⁹ provides music meta-data indexed by a unique compact-disc identifier. Attributes include album title, song titles, artist and genre. FreeDB is a volunteer effort; database entries are contributed by users and maintained by volunteers. It is possible for anyone to change entries that are already in the database (in order to correct mistakes), but there is no effort to verify

⁷More details regarding the evaluation procedure can be found at http://www.music-ir.org/mirex2005/index.php/{Audio_Genre_Classification,Audio_Artist_Identification}.

⁸More detailed results, including class confusion matrices and brief descriptions of each algorithm, can be found at <http://www.music-ir.org/evaluation/mirex-results/audio-{genre,artist}/index.html>.

⁹FreeDB is hosted online at <http://freedb.com>.

submission before they enter. FreeDB data (and server software) is freely downloadable under the GNU Public Licence¹⁰.

FreeDB accepts multiple entries for a given disc, perhaps to accomodate different markup conventions for compilations, re-releases, etc. The consequence is that some artists who have not produced many albums are nevertheless recognized as the artists of many albums in the database. Also, since records typically are not labelled with a genre, FreeDB contributors apply whatever genre labels they wish. Approximately 640 genres in FreeDB have been applied at least 50 times, though typographical errors make it difficult to estimate. To summarize, an artist may have many more FreeDB disc entries than his or her discography would suggest, and those entries may have different genre labels.

The parsing of the FreeDB database for the purpose of building a genre dataset was complicated because FreeDB is meant to be indexed by a compact-disc table of contents—there is no effort to standardize album titles across releases in different countries, and there is no effort to standardize artist names across albums. To obtain an index of the database by artist, the following algorithm was used. All artists with more than 10 [database] records were taken to be true artists (there were 20490 of these); all artists with between 2 and 10 such records were matched against the artists from the first set using a string-matching algorithm; when a suitable match was found the less-popular artist was merged to the more popular one, otherwise it was discarded; all artists with just 1 record were discarded; then all records without a genre label were discarded. Using the remaining records, a histogram over genres was built for each remaining artist. This set of histograms indexed by artist is what I will refer to as the **FreeDB** dataset. For convenience, I used a mysterious Levenstein-like measure created and implemented by Alexandre Lacoste. Alexandre's method aligned strings by local edits as well as block permutations. This made it possible to align string pairs such as **Mozart, W.A.** and **W.A Mozart**, and recognize the similarity between them.

3.5.1 Tags per Track

Figure 3.1 is a histogram of the number of tags per track. Even with log-scaling of the upper axis, there is a convex negative slope that indicates that the vast majority of songs are not labelled very many times. Roughly 100 artists have 40 discs, 500 artists have 20 discs, and 1100 artists have just 5. Few artists have fewer than 5 histogram points because of the parsing and filtering algorithm. On the other end of the graph, many artists have more than 100 database records with genre, and some have as many as 200.

3.5.2 Overall Tag Popularity

Figure 3.2 is a histogram of genre popularity—the total number of applications of each genre. This figure is log-scaled in both axes which gives a (roughly) linear downward slope between

¹⁰The GNU Public Licence guarantees freedom to use, modify and redistribute intellectual property. For more information, refer to www.gnu.org.

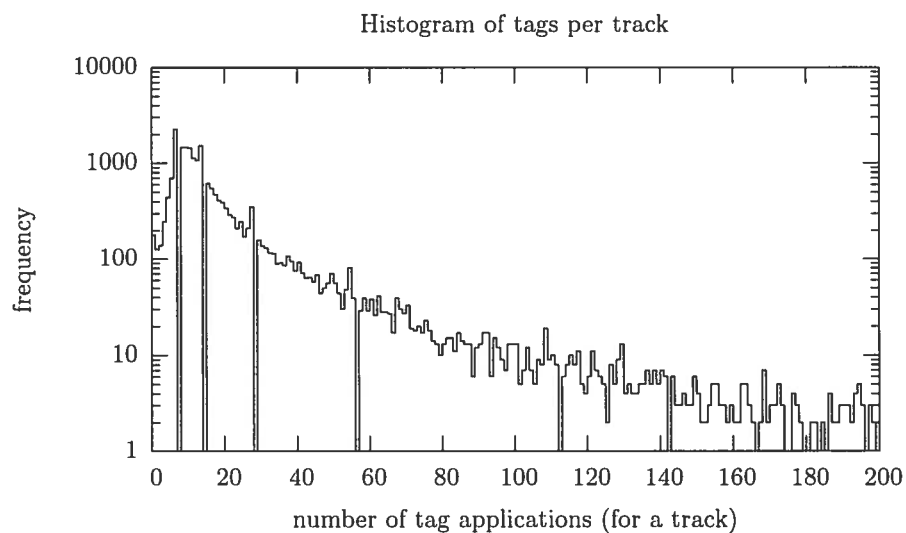


Figure 3.1. On the bottom axis, the ticks indicate the number of total labels (including repetitions) applied to an artist from FreeDB. On the upper axis, the ticks indicate how many artists had a given number of discs.

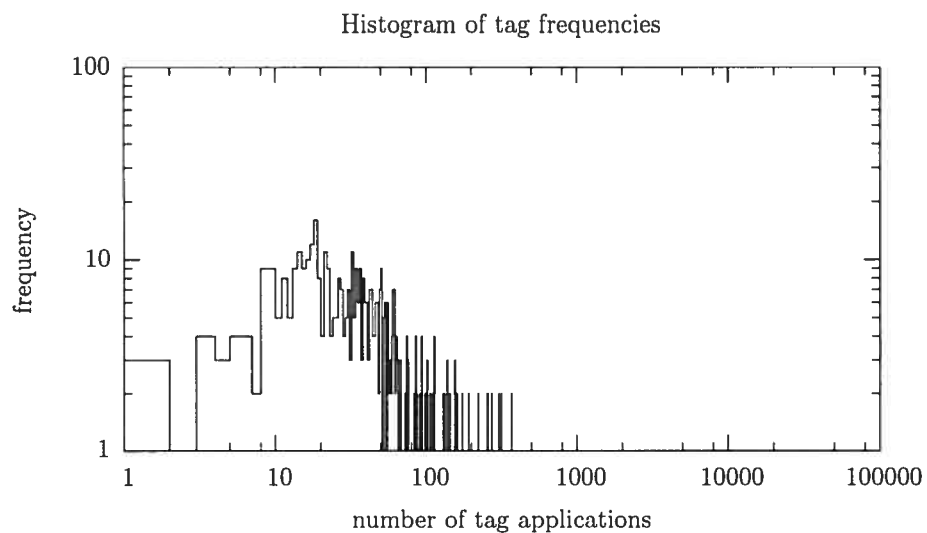


Figure 3.2. On the bottom axis, the ticks indicate the logarithm of the popularity of a genre over the entire FreeDB dataset. On the upper axis, the numbers indicate how many genres have a given (log) popularity. Note that log-scaling has been applied to both axes.

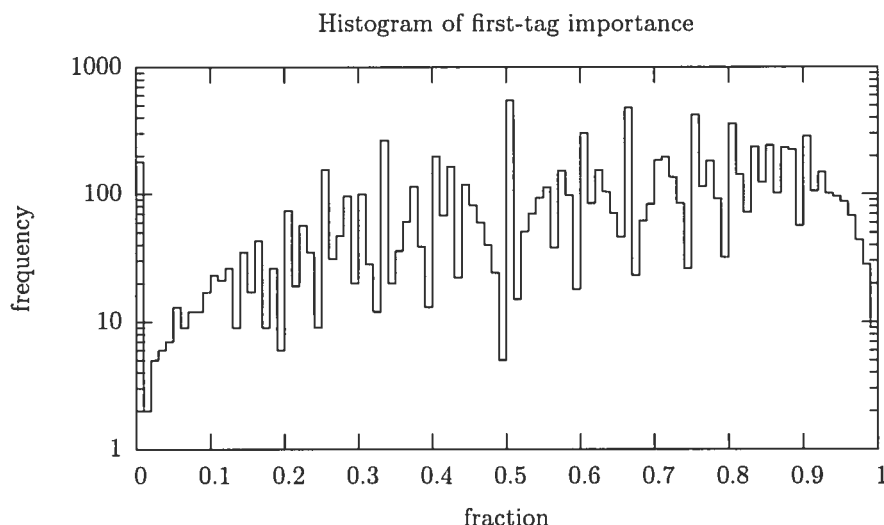


Figure 3.3. On the bottom axis, the ticks indicate the logarithm of the popularity of a genre over the entire FreeDB dataset. On the upper axis, the numbers indicate how many genres have a given (log) popularity. Note that log-scaling has been applied to both axes.

popularity 20 and 100. Most genres have been applied less than 50 times, though *rock* has been applied almost 100,000 times. This distribution is extremely sharp: there are just 8 genres that have been applied more than 10,000 times, 69 genres with popularity between 1000 and 10,000, about 120 genres with popularity between 100 and 1000, and 424 genres with popularity between 10 and 100

3.5.3 First-Tag Importance

Figure 3.3 is a histogram (over each track) of the fraction of label applications that went to the most popular label for the track. For example, if a track's most popular label had 4 applications (or votes), the second-most popular had 3, and another had 1, then the fraction that went to the most popular label would be $4/(4 + 3 + 1) = 0.5$. In Figure 3.3 there is a slight tendency for the fractions to be greater than 0.5, but a variety of fractions is common.

It is of particular relevance that these fractions are often small because if they had been large, then we would be able to discard non-leading labels without losing much information. This would have been convenient because there are a number of classification algorithms that would then have been applicable to the problem. Instead, we see in the histogram that discarding non-leading labels could well give us a very bad approximation of the tag distribution. It is not yet possible to be confident in that statement though, because there might be another basis for the space of genre tags that admits good approximations of track distributions using 1-hot vectors.¹¹ The search for such alternate representations is left for future work.

¹¹A 1-hot vector is non-zero in at most one dimension.

3.6 LastFM

Last.fm is an internet radio company that provides custom radio streams.¹² Their primary service is called “scrobbling”. Users permit Last.fm to accumulate information on their listening habits through media-player plugins for WinAmp¹³, iTunes¹⁴, XMMS¹⁵, etc, and in exchange Last.fm connects users with other users with similar taste, as well as new and interesting artists. With this service, Last.fm amassed a large community of users with a variety of listening habits. More recently they have expanded their services to allow another mechanisms of exploring music called tags, and they have taken advantage of their large listener community to collect a large number of tags for songs, albums and artists.

The **LastFM** data set is a set of <track, tag-histogram> pairs that have been tagged by one or more users who specifically chose to tag the track, and not the artist or album. In a version of the dataset provided in confidence to me this summer (July 2006) there were 66,191 pairs. For each song, only the most frequent (up to eight) tags are listed and each tag is given with its absolute frequency. A total of 30,408 tags occur at least once. This represents a dataset that is structurally similar to **FreeDB** but that is both cleaner and larger. Unfortunately, since it was only provided recently it is not the focus of this work.

¹²The Last.fm service is located online at <http://www.last.fm>.

¹³WinAmp is a popular program for playing mp3's on a Windows computer.

¹⁴iTunes is a popular program for playing mp3's on an Apple

¹⁵XMMS is a popular program for playing mp3's on a unix computer.

Chapter 4

Genre Classification of Recorded Audio

In section 2.1, I discussed what music genre is and raised some reasons for why one might want to predict it from audio. This chapter presents both known and novel ways of doing that, elaborating on experiments presented in Bergstra et al. [2006a].

In that article (Bergstra et al. [2006a]), we presented several algorithms for predicting music genre from audio. The primary method was to use the ensemble learner ADABOOST to select from a large set of audio features aggregated over short segments of audio. That method proved to be the most effective one in the genre classification contest at MIREX 2005, and the second-best method for recognizing artists. Furthermore, that article presented evidence that, for a variety of popular feature choices and classification algorithms, the technique of classifying these aggregated features over short audio segments was sensitive to the segment-length, which could be both too long and too short.

Although I was first author, Bergstra et al. [2006a] (attached as appendix C) was a collaboration. This chapter is entirely my own work, and follows directions of future work suggested in the article. For instance, the audio segmentation technique proposed in Bergstra et al. [2006a] was to partition the audio and classify each segment, which unnecessarily couples two variables. This chapter looks into whether it is the segment length that is important, or their number. This chapter also examines whether it is important that each segment be contiguous, or whether choosing an equivalent amount of audio randomly throughout a song is more effective. In another direction, Mandel and Ellis [2005b] suggest an alternative technique for summarizing each segment, and this chapter evaluates it against the feature set that won at MIREX 2005. Third, it has long been my suspicion that a particular popular feature set called the Mel-Frequency Cepstral Coefficients (**MFCC**, explained below) is not ideal for subsequent classification by a neural network; this chapter examines two alternative features that I call Mel-scale Phon coefficients and Mel-scale Sone coefficients.

The chapter is organized as follows. Section 4.1 describes several well-known audio features,

including their implementation details. Section 4.2 outlines previous work in song classification algorithms for genre. Section 4.3 describes two principles for song segmentation. Section 4.4 describes the proposed Mel-Scale Phon and Mel-Scale Sone coefficients, both in theory and in implementation. Section 4.5 describes two simple neural network topologies, and the learning algorithm used in section 4.6. Section 4.6 presents the performance of many combinations of feature, segmentation method, and classifier on the **Tzanetakis** problem (recall 3.1), and section 4.6.5 summarizes the findings.

4.1 Frame-level Features to Represent Music

The classification of recorded music presents several challenges that overwhelm a naive application of a vector classification algorithm. One challenge is that a raw music signal that is long enough to capture a recognizable instrument solo or rhythm has on the order of one hundred thousand dimensions. Another challenge is that classification should be invariant to imperceptible signal transformations such as slight acceleration, or shifting by a small number of samples. Another challenge is that small differences in small magnitudes of spectral components may be hidden in the raw waveform by energy in other frequencies but may be perceptually significant. Yet another challenge is that small overall pitch changes are not significant, but small changes in pitch of frequency components relative to one another are significant; the distinction between these is not evident in the raw waveform.

To overcome these challenges, it is conventional to apply feature-extraction algorithms to the raw audio before classifying it. Feature-extraction methods draw inspiration from a variety of sources: signal processing, physics of sound, psychoacoustics, speech perception, and music theory. Still, these features act more like ears than music critics; high-level concepts such as the lyrics, instrumentation, meter, and musical structure are beyond the analytic power of current feature-extraction methods. The features described in this section are simple mathematical transformations designed to capture some perceptually significant aspect of the sound. While each of these features is imperfect in the sense that imperceptibly different audio signals may yield significantly different feature vectors, still, for several standard learning algorithms, they ease the difficulty of generalizing genre decision boundaries from training examples. These transforms apply to *frames*, short segments of audio on the order of $\frac{1}{50}$ to $\frac{1}{20}$ of a second. In my implementation, a frame was defined to be 1024 samples at 22050kHz, corresponding to about 47ms of sound. Thus the Nyquist frequency was 11025 and when computing the DFT of each frame (recall 2.2.6) the frequency bands were spaced at 21Hz intervals. In my experiments, frames were used to partition the signal; they did not overlap, and there was no gaps between them.

overlap

4.1.1 FFTC

Many clues to the source or nature of a sound take the form of simple patterns that arise in the magnitude components of the DFT of a signal. For example, some percussive instruments like

cymbals and base drums tend to add energy at specific, sometimes even distinctive, frequencies. Other instruments that produce melodic tones are more difficult to identify, but tone quality of a melodic instrument is still related to amplitude patterns in \mathcal{F}_s .

$$\text{FFTC}[k] = |\mathcal{F}_s[k]| \quad (4.1)$$

The **FFTC** feature, defined in equation 4.1 from the signal's DFT, \mathcal{F}_s , was computed by first using FFTW¹ to perform a real-to-complex transform of each 1024-sample frame of the signal. FFTW does not normalize the transform, so that the absolute magnitude of each spectral component is proportional to the length of the frame used to compute it (as suggested by the definition of the DFT in equation 2.4). I considered this dependence on the framelen undesirable for feature extraction, so I scaled each output magnitude by $\frac{2.0}{\text{framelen}}$. This scaling factor meant that a full-strength input tone (e.g. a pure tone that attained the maximum recordable amplitude) was assigned intensity 1.0. The output of this transformation was a matrix in which each row has 512 complex elements, storing the DFT of each audio frame. The number of rows was proportional to the length of the recording.

The absolute value of each complex FFT output $a + bi$ was computed by the simple formula $\sqrt{a^2 + b^2}$. Each frame was thus reduced to a real-valued vector of 512 elements. In my experiments, I'll refer to an **FFTC** feature of 32 dimensions; this was obtained by using the first (lowest-frequency) 32 dimensions and discarding the rest.

4.1.2 RCEPS

Real Cepstral Coefficients (recall 2.2.7) are another popular feature developed for speech recognition. Perhaps the simplest way to compute them is to use the logarithm of $|\mathcal{F}_s|$ to map energy to loudness. While the logarithm is a crude estimation of loudness, it is computationally cheap. The requisite computations are expressed by equations 4.2 and 4.3 (culled from the Matlab signal-processing toolbox), in which L_{rceps} represents an intermediate computation, and a small constant γ prevents a logarithm of zero when silence is observed.

$$L_{\text{rceps}}[k] = \log(|\mathcal{F}_s[k]| + \gamma), \quad (4.2)$$

$$\text{RCEPS} = \text{dct}(\mathbf{L}_{\text{rceps}}) \quad (4.3)$$

In principle, a larger γ makes the response in L_{rceps} more linear with respect to signal intensity. In practice, classification performance was stable with $\gamma < \frac{1}{100}$, but deteriorated when γ was bigger. For my computations I chose $\gamma = \frac{1}{5000}$.

The logarithm and second DFT were performed on the magnitude DFT vectors themselves, so that after the second DFT, 256 real coefficients remained and I kept them all.

¹FFTW is a widely-used high-performance library implementing Fast Fourier transforms of real and complex signals in one or multiple dimensions. It is hosted online at <http://www.fftw3.org>.

```

matrix mel_warp()
    hz_nyquist = 11025, hz_spacing = 21,  hz_maxidx = 512
    mel_nyquist = mel(hz_nyquist),        mel_maxidx = 32

    W = zeros(hz_maxidx, mel_maxidx)
    for (h = 0; h < hz_maxidx; ++h)
    {
        m_idx = mel( h * hz_spacing ) / mel_nyquist * mel_maxidx
        j = floor(m_idx)
        W[ h, j + 1] =      ( m_idx - j )
        W[ h, j + 0] = 1.0 - ( m_idx - j )
    }
    normalize_col_sums( W, 1.0 ), return W

```

Figure 4.1. Pseudocode for the Mel Scale Transform of the DFT output, $|\mathcal{F}_s|$. Matrix-access boundary checking code has been omitted.

4.1.3 MFCC

Mel-frequency Cepstral Coefficients are computed almost identically to **RCEPS**, except that the input $|\mathcal{F}_s[k]|$ is first projected according to the Mel scale (Junqua and Haton [1996]). I approximated the Mel Scale with the one given in equation 4.4, in which the relation between frequency in Hertz (hz) and Mels (mel) is given formally. This curve was taken from the source code of HTK², and its correctness was confirmed by Dan Ellis in personal correspondence.

$$mel = 2595 \log_1 0(1 + hz/700) \quad (4.4)$$

If we introduce a linear transform M that implements the Mel-scale projection (see pseudocode in figure 4.1), Mel-Frequency Cepstral Coefficients can be defined using the notation used for the RCEPS (see equations 4.5 and 4.6). The normalization function at the end of the pseudo-code in figure 4.1 scales the columns of W so that each Mel bin receives a constant amount (1.0) of energy from its contributing DFT bins.³

$$L_{mfcc}[k] = \log(M(|\mathcal{F}_s|)[k] + \gamma) \quad (4.5)$$

$$MFCC = \text{dct}(L_{mfcc}) \quad (4.6)$$

MFCCs have been used with success for measuring music similarity by, for example, Logan [2000], Tzanetakis et al., Tzanetakis and Cook [2002], West and Cox [2005], and Mandel and Ellis [2005b].

²Hidden Markov Model Toolkit (HTK) is hosted online at <http://htk.eng.cam.ac.uk/>.

³Numerous subtle tricks are generally used to tweak MFCC computation to improve performance. For example, loudness curves might be applied to pre-filter the signal and the signal may be scaled non-uniformly along its duration beforehand. Matlab software on Dan Ellis' website (<http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/>) offers many flavours of MFCC in addition to other audio transforms for speech recognition.

```

int zero_crossing_count(double * s, int length)
long int dbl_sign = (1 << 63)
const long int * t = (long int *) s
int was_neg = t[0] & dbl_sign
sign_changes = 0
for (i = 0; i < length(s); ++i)
{
    is_neg = t[i] & dbl_sign
    sign_changes += (is_neg XOR was_neg)
    was_neg = is_neg
}
return sign_changes

```

Figure 4.2. Count the zero-crossings of floating-point signal s without floating-point math.

In my implementation, I mapped the 512 DFT bins to 32 bins, so that after the discrete cosine transform only 16 coefficients remained from the 1024-sample audio frame. This vector was transformed by a Mel-scale warping matrix W , defined using equation 4.4 and the pseudocode in figure 4.1.

4.1.4 ZCR

The **Zero-Crossing Rate** is the rate of sign-changes along the signal. This feature has been used in both speech and music feature extraction, and is defined formally in equation 4.7, in which T is the length of the signal s , and the indicator function I_{cond} is 1 if $cond$ is true and 0 otherwise. In a signal with a single pitched instrument, the **ZCR** is correlated with the dominant frequency Kedem [1986]. When several instruments are combined, it is not clear how to interpret the **ZCR**.

$$ZCR = \frac{1}{T} \sum_{t=0}^{T-1} I_{s[t]s[t-1] < 0} \quad (4.7)$$

ZCR was used by Tzanetakis et al. and Tzanetakis and Cook [2002].

The **zero-crossing rate** was computed efficiently in the time domain with the algorithm given in figure 4.2. This algorithm is much faster than several obvious implementations that use floating-point arithmetic on an x86 CPU.

4.1.5 Rolloff

The **rolloff** feature is the a -quantile of the total energy in $|\mathcal{F}_s|$. It is the frequency under which fraction a of the total energy is found. If K is the Nyquist frequency of our signal, then the **rolloff** feature is defined by equation 4.8.

$$ro(a) = \max_y \left\{ y : a > \frac{\sum_{k=0}^y |\mathcal{F}_s^{(k)}|}{\sum_{k=0}^K |\mathcal{F}_s^{(k)}|} \right\} \quad (4.8)$$

Rolloff was used to classify speech versus music by Scheirer and Slaney [1997], as well as Tzanetakis et al..

4.1.6 Spectral Centroid, Spectral Spread

The **spectral centroid** and the **spectral spread** are the sample mean and the sample variance, respectively, of $|\mathcal{F}_s|$ (considered as a histogram, and normalized to sum to 1). They are defined in equations 4.9 and 4.10.

$$\text{CENT} = \frac{\sum_k k |\mathcal{F}_s[k]|}{\sum_k |\mathcal{F}_s[k]|} \quad (4.9)$$

$$\text{SPREAD} = \frac{\sum_k |\mathcal{F}_s[k]| (k - r_{\text{cent}})^2}{\sum_k |\mathcal{F}_s[k]|} \quad (4.10)$$

These features were used by Scheirer and Slaney [1997], as well as Tzanetakis et al..

4.1.7 Autoregression (LPC, LPCE)

The k linear predictive coefficients of a signal s are a low-dimensional representation of the comportment of a signal. They are defined in equations 4.11 and 4.12. These can be computed efficiently from the signal's autocorrelation by the Levinson-Durbin recursion (Makhoul [1975]).

$$\text{LPC}(k) = \arg \min_{a \in \mathbb{R}^k} \sum_{t=1}^T \left(s[t] - \sum_{i=1}^k a[i] s[t-i] \right)^2 \quad (4.11)$$

$$\text{LPCE}(k) = \min_{a \in \mathbb{R}^k} \sum_{t=1}^T \left(s[t] - \sum_{i=1}^k a[i] s[t-i] \right)^2 \quad (4.12)$$

These were used by Ahrendt et al. [2004].

Feature Summary

In this section I've presented several types of frame-level feature, but there are many other examples of frame-level features (eg Tzanetakis and Cook [2002], Li and Tzanetakis [2003], Bello et al. [2005], Lidy and Rauber [2005]) as well as feature extraction methods designed specifically for finding long-timescale structure in music, such as the beat histogram (Tzanetakis and Cook [2002]) and the autocorrelation phase matrix (Eck and Casagrande [2005]). A survey by Aucouturier and Pachet [2003] includes a list of features used for music information extraction, but research continues (e.g. Bello et al. [2005], Lidy and Rauber [2005]). Nevertheless, the ones I've presented here have been shown to be effective (Bergstra et al. [2006a], Mandel and Ellis [2005a]) and they are the ones I use in my own experiments below. The program I used to compute these features is briefly documented in appendix A.1.

4.2 Song Classification Algorithms

Whereas the features described in section 4.1 are extracted for each frame, the genre or artist labels are fixed for a whole song, whose length is not known in advance. This section describes how several authors have combined feature-extractors and multiclass vector-classifiers into song classification algorithms, by concentrating on different temporal scales.

4.2.1 No Aggregation

Xu et al. [2003] use SVMs directly on frame-level features to post good results on a small in-house dataset. West and Cox [2005] also classify individual frames by Gaussian mixtures, LDA, and decision trees, but note a significant improvement when they use the mean and variance of previous frames. Labelling each frame independently is inefficient from both a computational and a statistical point of view. Computationally, there are so many frames (e.g. 20 per second) that even classifying them takes a long time. Statistically, it's inefficient because temporally close vectors are highly correlated.

4.2.2 Feature Sample Statistics

Lambrou et al. [1998] overcame the problem of variable-length songs by summarizing the feature stream of an entire song by feature dimension-wise sample mean and sample variance. This technique has also been used by Tzanetakis and Cook [2002] and Li and Tzanetakis [2003]. More recently, Mandel and Ellis [2005b] included the entire sample covariance. This addition, using an SVM with a kernel related to the symmetric KL divergence between the distributions parametrized by the mean and covariance, led to good performance at MIREX 2005's genre competition.⁴

4.2.3 Song Segmentation

Several authors have collected sample statistics from multiple song portions (segments) rather than the whole (eg: Lidy and Rauber [2005], West and Cox [2005], Bergstra et al. [2006a]). The segments are classified individually and vote for the song label. West and Cox [2005] segmented the song using an algorithm that finds note boundaries, while Bergstra et al. [2006a] and Lidy and Rauber [2005] segmented the song using pre-determined temporal boundaries.

4.2.4 Mixture Modelling

Logan and Salomon [2001], Aucouturier and Pachet, Pampalk et al. [2005b], and Mandel and Ellis [2005b] fit the set of frame-level features with mixtures of Gaussians. Logan and Salomon [2001] fit the mixture components by K-means clustering (explained in chapter 4 of Duda et al. [2001]), and defined the distance between songs as the Earth-Mover's Distance (Rubner et al. [2000]) Aucouturier and Pachet, Pampalk et al. [2005b] and Mandel and Ellis [2005b] used expectation

⁴The symmetric KL divergence between distributions P and Q , is the average of $KL(P||Q)$ and $KL(Q||P)$.

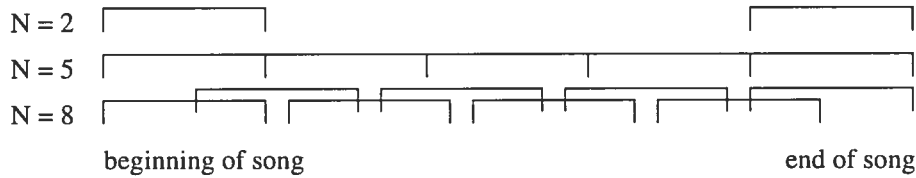


Figure 4.3. Three contiguous segmentations of a song, in which $F = 5T$. In the first case where $N = 2$, the segments are aligned with the beginning and end of the song. When $N = 5$ we see that the segments partition the song. When $N = 8$, we see the segments overlap.

maximization (Bishop [1996]) and defined the distance between songs as the Symmetric Kullback-Leibler divergence.

In theory, Gaussian mixtures can provide at least as good a fit to any empirical distribution compared with a single Gaussian, but this theoretical benefit comes at the price of two important practical advantages. Firstly, there is no analytic or convex procedure for estimating the parameters of a Gaussian mixture, so that smart heuristics are necessary to ensure that multiple Gaussians fit even as well as one. Secondly, multidimensional numerical integration is necessary to estimate the symmetric KL divergence between Gaussian mixtures, though other divergences can be evaluated analytically. (This frustration comes from the fact that the KL puts the sum of contributions from mixture components inside a logarithm function.)

4.2.5 Rhythm, Temporal Dynamics

Other authors have recognized that treating frame-level features as independent identically-distributed samples from a stationary distribution makes it impossible to recognize patterns (e.g. rhythm) on longer time-scales. The distinction between feature-extraction and feature-aggregation is blurred in the case of certain rhythm features. For example, Lidy and Rauber [2005] present a classification algorithm whose features are based on envelopes of narrow-band filtered versions of the signal. Meng et al. [2005] and Ahrendt et al. [2004] use linear auto-regression coefficients to summarize the sequence of frame-level features, as opposed to the mean and covariance. This method has the advantage of being almost as cheap to compute as the mean and covariance, while capturing some of the temporal dynamics in the features of the signal. Combining the possibility of using temporal dynamics with the technique of segment classification (4.2.3) Ahrendt and Meng [2005] partition each song into 1.2s segments, and apply an auto-regressive model to the MFCC dimensions.

4.3 Contiguous and Non-Contiguous Song Segmentation

In this section I describe two generalizations of the audio segmentation technique proposed in Bergstra et al. [2006a]. Briefly, that technique was to partition (no gaps, no overlap) the audio into segments of a predetermined length, and classify each segment separately before voting for the entire song.

One generalization of the partition segmentation method is to decouple the segment length and the number of segments. Instead of partitioning the song, we simply give the song a uniform (qualitatively) covering of segments. More formally, suppose we were to partition a song into N blocks of T frames. If the song has F frames in total, then the first frame of the n^{th} segment is $F_0 = \lfloor \frac{n-1}{N-1} (F - T) \rfloor$. The frames of the segment are $\{F_0, F_0 + 1, \dots, F_0 + T - 1\}$. This method is illustrated in figure 4.3.

A more extreme generalization of the partition segmentation method (indeed of the contiguous segmentation method) is to relax the constraint that each segment be temporally contiguous. It may be that the partition segmentation method works by the principle of data amplification (Caruana [1997]) or Bootstrap-style resampling (Efron and Tibshirani [1994]) and simply a number of estimates of the sample statistics using different data resamples will achieve the same effect. The name *segment* is a misnomer in this segmentation method because every segment can have samples from anywhere—the segments themselves are identically independently distributed. This method permits segments of any length (even longer than the song) and we can obtain as many of them as we like.

4.4 Mel-Scale Phon and Sone Coefficients

In this section I introduce two features that are simplified versions of the **MFCC**. They are attempts to improve the **MFCC** as an input to a music classification algorithm, especially in the case that the vector-classifier performs a linear transform of its input.

Recall that the final transform in the computation of **MFCC** is the DCT, which is a linear transform. Recall also that many classification learning algorithms construct linear transforms of their input as part of the classification process. In such cases, it is not necessarily wise to apply the DCT and deprive the learning algorithm of significant information (half the basis of the original space is discarded). If it were clear that the DCT removes un-informative dimensions of input, then its application would be justified; this chapter investigates whether this is the case. The **Mel-scale Phon Coefficients (MPC)**, defined in equation 4.13 (using the same frequency projection M from section 4.1.3), are simply the **MFCC** with the final discrete cosine transform omitted.

$$\text{MPC}_i := \log(\gamma + (M \cdot |\mathcal{F}(s)|)[i]) \quad (4.13)$$

The Sone scale is an alternative to the Phon scale, in which the loudness is computed as the cube-root of the signal energy. In equation 4.14 I define the **Mel-scale Sone Coefficients** in exactly the same way as the Phon coefficients, except that the logarithm is replaced by the cube-root (denoting by m_i , the i^{th} row of M).

$$\text{MSC}_i := (m_i \cdot |\mathcal{F}(s)|)^{\frac{1}{3}} \quad (4.14)$$

In both **MPC** and **MSC** features, each dimension is a positive scalar that is supposed to be proportional to the loudness in a particular frequency range. The ranges are chosen to be equally-spaced pitches.

4.5 Neural Network Classifier

In this section I present the implementation details of my neural network. I implemented two simple neural network topologies to serve as vector-space classifiers: NN_{direct} and NN_{hid} .

NN_{direct} was a very simple multiclass neural network. It implements the function given in equation 4.15, in which W is a matrix whose size matches that of the feature set and the 10 output classes, and b is a vector of length 10.

$$NN_{direct}(x; W, b) := \text{softmax}(Wx + b) \quad (4.15)$$

NN_{hid} was a slightly more complicated variant, with a sigmoidal hidden layer. I chose to use 24 hidden nodes as a compromise between network capacity and computational cost. In preliminary trials I used various numbers of hidden units up to 256 without noticing any consistent improvement in performance. It implements the function given in equation 4.16, in which V is a matrix representing a linear transform from \mathbb{R}^{24} to \mathbb{R}^{10} , c is a vector of length 10, b is a vector of length 24 and W is a matrix that maps the input features to \mathbb{R}^{24} .

$$NN_{hid}(x; W, V, b, c) := \text{softmax}(c + V \text{logistic}(b + Wx)) \quad (4.16)$$

The optimization of these networks was done by gradient descent using the delta-bar-delta method, which was observed to be faster than any of the gradient-descent methods implemented in the GSL.⁵ The delta-bar-delta method was implemented as described in figure 4.4.

All weights of the network (V , W , b , c) were initialized to be uniform on ± 0.05 , regardless of the number of input dimensions. All input dimensions were shifted and scaled to have mean 0 and variance 1 over the training set. Optimization was stopped when one of two conditions was met: 600 iterations had completed, or no validation set improvement had been found on iteration $i > 50$ since iteration $i/2$. The validation set was a random 20% of the songs in the training set.

In Bergstra et al. [2006a] a model-averaging scheme was employed to integrate over the randomness introduced by the network initialization. Model-averaging was not done in these experiments due to both insufficient computational time and the observation that fold-variance was already low relative to the differences in performance caused by other variables under investigation. Bagging (Breiman [1996]) was deliberately not done because it would interfere with the data-amplification methods under investigation.

4.6 Feature and Segmentation Performance

In this section I evaluate four features, two amplification methods, 4 segment lengths, 4 numbers of segments and the two classifiers described above on the **Tzanetakis** dataset. Accuracy scores

⁵The GNU Scientific Library (GSL) is a numeric library under the GNU Public Licence. Its homepage is <http://www.gnu.org/software/gsl>. It implements Polak-Ribiere and Fletcher-Reeves conjugate gradient descent and the Broyden-Fletcher-Goldfarb-Shanno quasi-newton method, though there has been sporadic discussion on the gsl mailing list challenging the efficiency of the line search in these implementations.

```

gsl_vector delta_bar_delta (
    gsl_vector x,          //best known value's point
    double f,              //best known value
    gsl_vector gradient    //gradient at best point
)
{
    static gsl_vector lr    // dimension-wise learning rate
    static step             // global learning rate

    gsl_vector x1           // probe point
    gsl_vector g1           // gradient at probe point

    step /= 0.5
    do {
        step *= 0.5
        dx = (-step / norm2 (gradient) * gradient) .* lr
        x1 = x + dx
        [f1,g1] = func_to_minimize(x1)

        for (i = 0; i < size(lr); ++i) {
            if ( gradient[i] * g1[i] > 0 ) {
                lr[i] += ( lr[i] < 10.0 ) ? 0.05 : 0.0
            } else {
                lr[i] *= ( lr[i] > 0.1 ) ? 0.95 : 1.0
            }
        }
    } while (f1 > f0)

    state->step = step * 1.1
    return [x1,g1]
}

```

Figure 4.4. Delta-Bar-Delta pseudocode. The algorithm maintains both a dimension-wise learning rate and a global learning rate. The dimension-wise learning rates are constrained to a range around 1.0, while the global learning rate is un-constrained. The magnitude of the gradient is ignored.

are estimated by 5-fold cross-validation. Despite the fold-variance being optimistically biased, it is used to compute a biased 90% confidence interval for the mean accuracy. The interval was computed using the equation in 4.17, in which the constant 2.132 comes from the Student-T distribution, with 4 degrees of freedom at 95% (left-cumulative).

$$mean_{err} = 2.132 \frac{\sqrt{Var}}{\sqrt{5}} \quad (4.17)$$

I chose the four segment lengths to be 540, 180, 60, and 20 frames, and the four numbers-of-segments to be 1, 3, 9, and 27. Clips in **Tzanetakis** are all roughly 30s long. There is some variation in their lengths, so I used only the first $2^{19} + 2^{17}$ samples of each clip, which corresponds to approximately 29.7 seconds of audio. This number of samples corresponds to 640 frames of 1024 samples. I chose to evaluate segment lengths that were obtained by repeatedly dividing this total by 3. In this way, the longest segment corresponds to roughly 24 seconds, the shortest segment corresponds to just under 1 second, and the segment lengths are equally spaced on a logarithmic scale.

I chose four features: **MFCC.MCV**, **MIREX.MV**, **MSC.MCV**, and **MPC.MCV**. The **MFCC.MCV**, **MSC.MCV**, and **MPC.MCV** segment features were calculated by computing sample mean and covariance (lower triangle) over the frames of a segment. MFCCs were calculated at a resolution of 16 coefficients per frame, yielding an **MFCC.MCV** feature of $16 + 16 * (16 + 1)/2 = 152$ dimensions, while the **MSC.MCV** and **MPC.MCV** features had a resolution of 32 coefficients per frame, yielding $32 + 32 * (32 + 1)/2 = 560$ dimensions.

The **MIREX.MV** was the same one from our winning MIREX05 competition entry; it was the means and variances (not covariance) of a larger number of features (summarized in table 4.1). For the **MIREX.MV** feature, MFCCs were calculated in a manner similar, but not identical to the calculation for **MFCC.MCV**. I changed the *mel_maxidx* variable in the pseudocode of figure 4.1 to 128. This larger vector allowed me to compute 64, rather than 16 **MFCC** coefficients after the second DFT. The **Rolloff** feature was computed at 16 equally-spaced intervals on the Hz scale. For each frame a 402-dimensional feature vector was computed by concatenating 256 RCEPS, 64 MFCC, 32 LPC, 1 LPCE, 32 FFTC, 16 rolloff, and 1 zero-crossing rate. **MIREX.MV** involved a total of 402 frame-level features, so that the means and variances had $402 \times 2 = 804$ dimensions.

Feature	RCEPS	MFCC	LPC	LPCE	FFTC	Rolloff	ZCR
Dimensionality	256	64	32	1	32	16	1

Table 4.1. A summary of the features in MIREX.MV. Each of these is represented by the mean and variance.

4.6.1 Mel-scale Cepstral Coefficients

The classification performance using **MFCC.MCV** is given in figure 4.5. When using contiguous segmentation, performance using NN_{direct} ranged from 0.453 in the case of a single segment of

20 frames, to 0.709 in the case of 27 segments of size 60. Performance using NN_{hid} ranged from 0.421 in the case of a single segment of 20 frames, to 0.754 in the case of 27 segments of size 20. When using non-contiguous segmentation, performance using NN_{direct} ranged from 0.447 in the case of a single segment of 20 frames, to 0.656 in the case of 27 segments of size 60. Performance using NN_{hid} ranged from 0.470 in the case of a single segment of 20 frames, to 0.768 in the case of 27 segments of size 60.

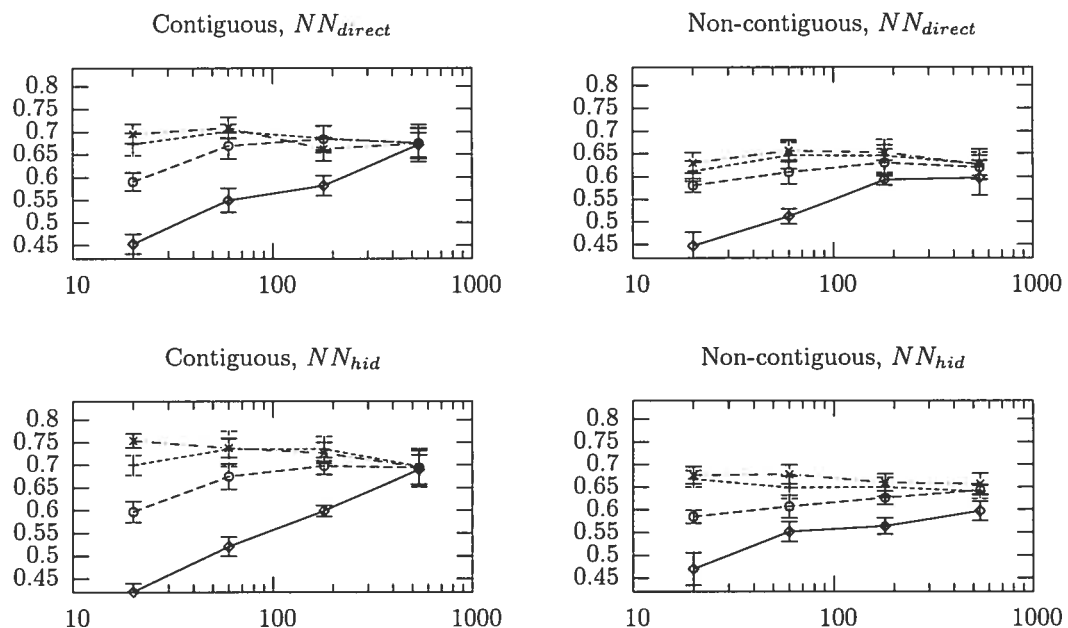


Figure 4.5. The performance of the MFCC.MCV features. The solid line corresponds to the 1-segment classifier, the long dashes to 3, the short dashes to 9, and the '-.' to 27. The y-axis is test-set classification accuracy. The x-axis is segment-length, in 44-ms frames.

4.6.2 MIREX.MV

The classification performance on **MIREX.MV** is given in figure 4.6. When using contiguous segmentation, performance using NN_{direct} ranges from 0.543 in the case of a single segment of 20 frames, to 0.748 in the case of 27 segments of size 20. Performance using NN_{hid} ranges from 0.510 in the case of a single segment of 20 frames, to 0.786 in the case of 27 segments of size 20. When using non-contiguous segmentation, performance using NN_{direct} ranges from 0.575 in the case of a single segment of 20 frames, to 0.707 in the case of 27 segments of size 20. Performance using NN_{hid} ranges from 0.570 in the case of a single segment of 20 frames, to 0.730 in the case of 27 segments of size 20.

In general the best performance with **MIREX.MV** is obtained using a large number of short segments. As the segments were allowed to overlap, and as fewer of them were used, performance deteriorated.

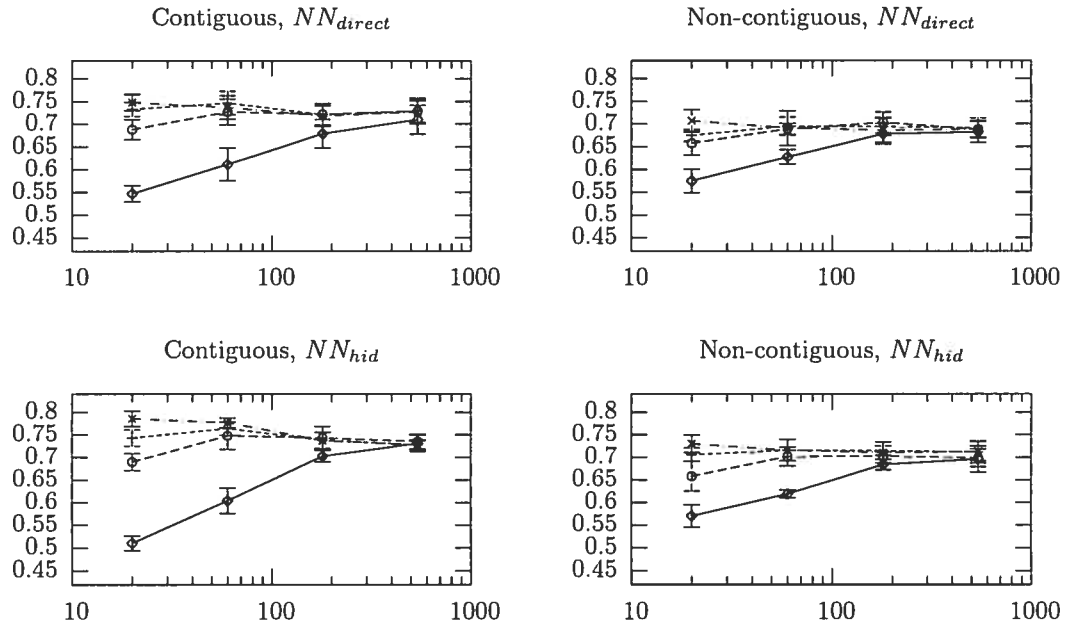


Figure 4.6. The performance of the MIREX.MV features. The solid line corresponds to the 1-segment classifier, the long dashes to 3, the short dashes to 9, and the '-.' to 27. The y-axis is test-set classification accuracy. The x-axis is segment-length, in 44-ms frames.

4.6.3 Mel-scale Phon Coefficients

The classification performance using MPC.MCV is given in figure 4.7. When using contiguous segmentation, performance using NN_{direct} ranges from 0.533 in the case of a single segment of 20 frames, to 0.785 in the case of 27 segments of size 60. Performance using NN_{hid} ranges from 0.501 in the case of a single segment of 20 frames, to 0.812 in the case of 27 segments of size 60. When using non-contiguous segmentation, performance using NN_{direct} ranges from 0.537 in the case of a single segment of 20 frames, to 0.655 in the case of 27 segments of size 60. Performance using NN_{hid} ranges from 0.485 in the case of a single segment of 20 frames, to 0.674 in the case of 27 segments of size 60.

4.6.4 Mel-scale Sone Coefficients

The classification performance using MSC.MCV is given in figure 4.8. When using contiguous segmentation, performance using NN_{direct} ranges from 0.540 in the case of a single segment of 20 frames, to 0.795 in the case of 27 segments of size 60. Performance using NN_{hid} ranges from 0.494 in the case of a single segment of 20 frames, to 0.809 in the case of 27 segments of size 60. When using non-contiguous segmentation, performance using NN_{direct} ranges from 0.526 in the case of a single segment of 20 frames, to 0.700 in the case of 9 segments of size 60. Performance using NN_{hid} ranges from 0.522 in the case of a single segment of 20 frames, to 0.722 in the case

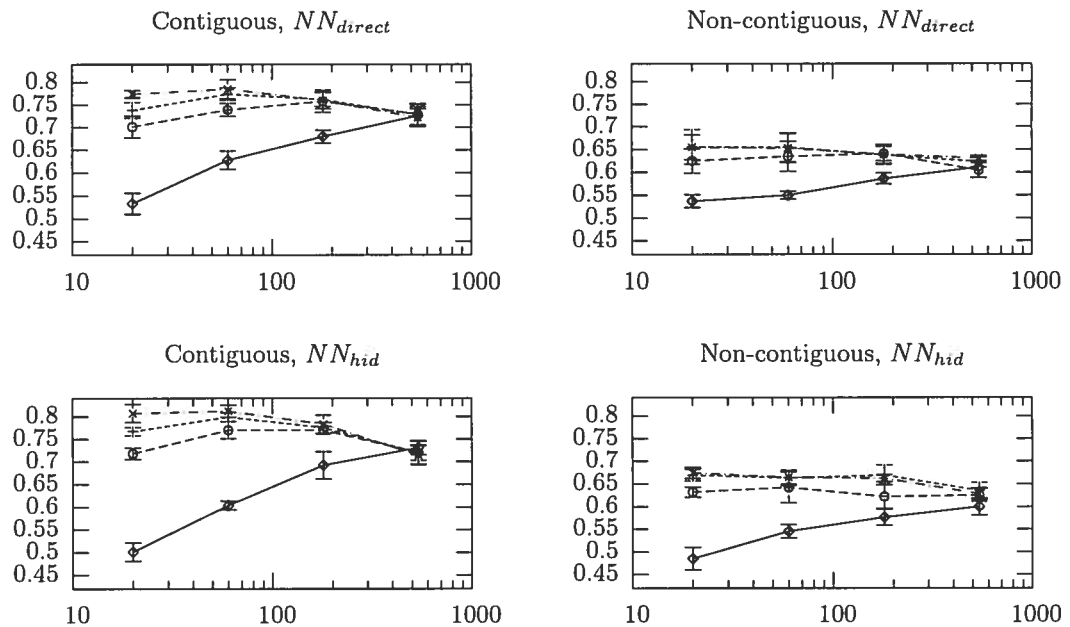


Figure 4.7. The performance of the MPC.MCV features. The solid line corresponds to the 1-segment classifier, the long dashes to 3, the short dashes to 9, and the '-' to 27. The y-axis is test-set classification accuracy. The x-axis is segment-length, in 44-ms frames.

of 27 segments of size 20.

4.6.5 Analysis

With regards to the segment length and count, the best classification rates were consistently found using many short segments and the worst were consistently found few short segments. The difference between best and worst was consistently 25-30%. Variance in the estimate of the mean performance makes it impossible to pronounce a clear winner between 20-frame segments and 60-frame ones, though 60-frame segments appeared to be slightly ahead. Using longer segments of 180 or 540 frames tended to make the number of segments less influential. In some cases (e.g. **MFCC.MV**, NN_{hid}) the accuracy appeared to rise without bound as the segment length decreased and the segment count increased. It is left for future work to explore further along this curve to see how high the accuracy can go.

With regards to the segmentation method, the contiguous version was superior in almost every case. Only in the case of a single 20-frame segment was non-contiguous segmentation better. In fact, non-contiguous segmentation was so bad that a single long contiguous frame consistently outperformed every type of classification based on non-contiguous frames. Musically, this is an encouraging result: although genres can largely be predicted from simple features, at least the order of frames is important. Statistically, this is a surprising result: it suggests that in several audio feature spaces based on means and (co-)variances, the classes are so delicately separated

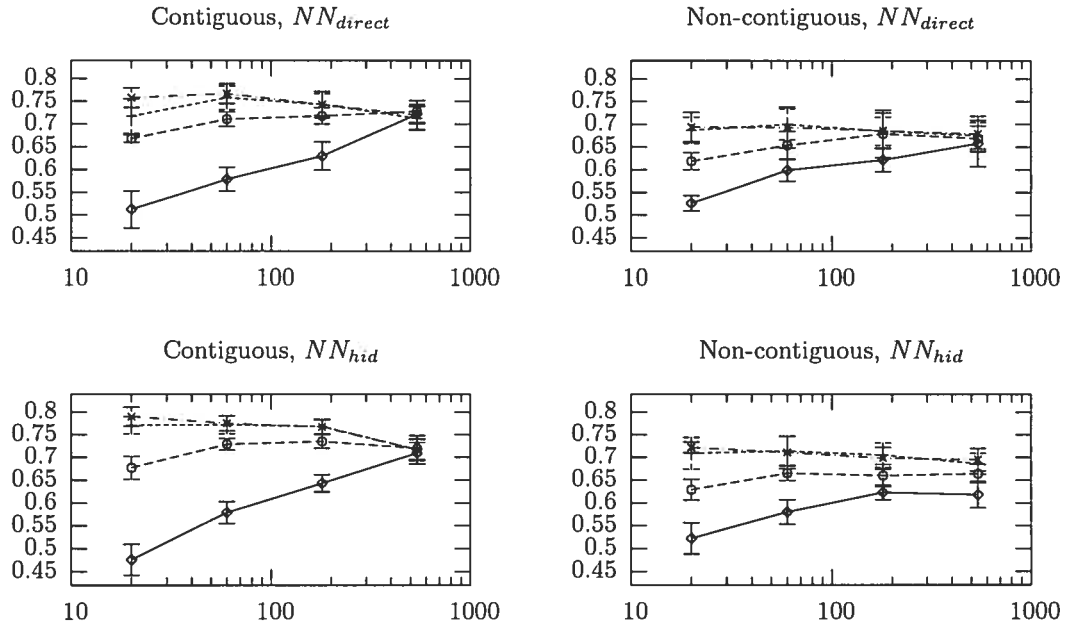


Figure 4.8. The performance of the MSC.MCV features. The solid line corresponds to the 1-segment classifier, the long dashes to 3, the short dashes to 9, and the '-.' to 27. The y-axis is test-set classification accuracy. The x-axis is segment-length, in 44-ms frames.

that frame-resampling blurs them significantly.

With regards to the choice of feature, **MPC.MCV** and **MSC.MCV** performed better than **MFCC.MCV** and **MIREX.MV**. The results suggest that **MPC.MCV** is perhaps a little better than **MSC.MCV** but their scores were too close to declare one superior: the best 5-fold mean was obtained using NN_{hid} on **MPC.MCV** (accuracy of 0.812 ± 0.013), but the second-best mean was obtained from NN_{hid} on **MSC.MCV** (accuracy of 0.809 ± 0.015). The best performances on the **MFCC.MCV** and **MIREX.MV** trailed behind with accuracies of 0.754 ± 0.015 and 0.786 ± 0.017 respectively. It is clear that for the two classifiers I used, the discrete cosine transform at the end of the MFCC calculation is damaging. Still, the DCT is a good dimensionality reduction in this task. It halves the feature length and reduces the covariance size by a factor of 4, while incurring a relatively small penalty in classification accuracy (19% error vs 25%).

With regards to the runtime of these algorithms, the computational cost can be understood by considering the steps of the algorithm separately. Feature extraction of **MIREX.MV** from the entire database of 30 000 seconds of audio took about 3 minutes. Segmentation, and the computation of covariance matrices for **MSC.MCV** was negligible. The gradient descent to fit neural network parameters had a cost proportional to both the number and length of training examples; in the case of a single segment of **MFCC.MCV** training took around one minute, while in the case of **MIREX.MV** and 27 segments, training took around half an hour.

4.7 Discussion

These experiments have answered some of the questions raised by Bergstra et al. [2006a] and further confirmed the importance of segmentation and feature extraction. Consistent with Bergstra et al. [2006a], I find that segments longer than 3 seconds are suboptimal and that short segments such as 1 or 3 seconds seem about equally good. The method of contiguous segmentation that allows for (a greater number of) overlapping segments is an improvement over the partitioning method, but it is not just having more segments that matters because the non-contiguous segmentation method failed even to match the performance of a single long segment. Bergstra et al. [2006a] showed that the mean and variance in MFCC features is not as predictive of genre as the **MIREX.MV** set, and this chapter finds (using two simple neural network classifiers) that the mean and covariance in MFCCs are better than the mean and variance, but still inferior to **MIREX.MV**. However, both new Mel Scale Features (**MPC.MCV**, **MSC.MCV**) worked better than the **MIREX.MV** set, even though they use fewer dimensions (560 vs. 804).

As for the raw performance achieved, the scores found here are not the highest published on **Tzanetakis**. Trials performed using Norman Casagrande's implementation of ADABOOST used in Bergstra et al. [2006a] achieved 83%, edging out the best here (81.2%). Still, both of these scores compare favourably with previous classification rates on this dataset, eg Li and Tzanetakis [2003] (71%), and Tzanetakis and Cook [2002] (61%).

Chapter 5

Genre Classification of FreeDB Histograms

The article from which this chapter was derived will appear in the proceedings of Tzanetakis [2006] as Bergstra et al. [2006b]. In this chapter, I look at one aspect of the relationship between AllMusic and FreeDB as sources of genre information. Recall that AllMusic (described in section 3.2) is, among other things, an online reference for the genre (as taxonomy) of artists. In this chapter I introduce a heuristic method for distilling a sort of artist genre (as distribution) from the FreeDB database. One interesting question about this FreeDB-genre regards the relation between FreeDB-genre and the more accepted AllMusic notion of genre. This chapter presents experimental evidence that suggests a simple injective map from FreeDB-genre to AllMusic-genre is relatively accurate, and thus that FreeDB genre contains at least as much information as AllMusic genre. This is good news because FreeDB-genre constitutes a freely-available alternative reference for genre and music similarity.

In order to quantify one aspect of the relationship between AllMusic-genre and FreeDB-genre, I chose a small sample of artists in FreeDB. I tabulated the FreeDB-genre of these artists and looked up their AllMusic-genre online, and used a neural network to predict the latter from the former. This section describes the details of the procedure, and presents the results of how well it worked.

5.1 Input: FreeDB

The entire FreeDB database is available from <http://www.freedb.org>. Although the database is large (roughly 8GB in the Unix distribution) it was distilled to a manageable size by grouping entries with the same artist (according to an approximate string-matching of artist fields) and trimming the rarest artists. After trimming artists with fewer than 10 disc entries, 20 470 artists remained. After trimming artists with fewer than 50 disc entries, 2 388 artists remained. We selected 500 artists at random from the top 2 388 as the subjects of our experiment. There are 639

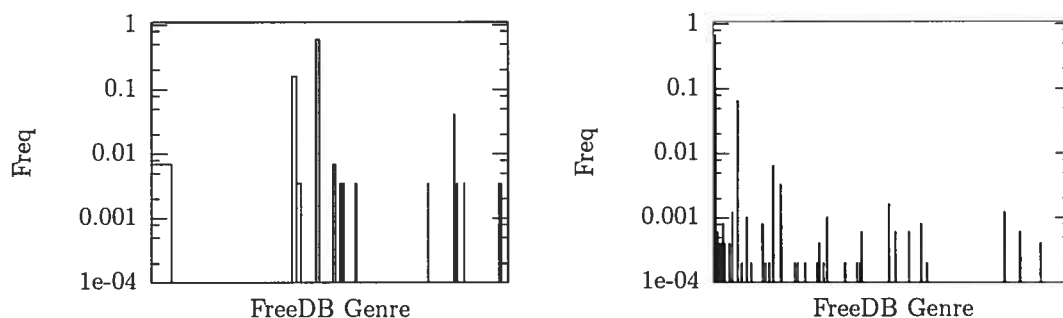


Figure 5.1. On the left is the FreeDB-genre histogram of 2Pac (log-scaled y-axis). The histogram is dominated by *Rap* and *HipHop*. On the right is the FreeDB-genre histogram of Mozart, W.A. (log-scaled y-axis). The distribution takes a peak at *Classical*. In both cases note that many genres have been applied.

genres in FreeDB that occur in at least 10 album labels, though only 408 of them are mentioned among the albums of the 500 artists chosen. The most popular of these genre labels are *Rock*, *Classical*, and *Pop*, while the less-frequently used genre labels span a wide space of possibilities (eg. *Weihnachtslieder*, *Hungarian Folk*, *Viking Metal*). In this way, each artist was mapped to a FreeDB-genre that was a probability distribution over 408 labels, derived from a histogram over at least 50 disc entries. Some sample histograms are shown in figures 5.1.

5.2 Target: AllMusic

For each artist chosen from FreeDB, the corresponding AllMusic entry was retrieved via the web interface at (<http://allmusic.com>). The genre associated with each artist was recorded, and when multiple genres were associated with an artist (which happened in the case of classical composers), all were recorded. Out of the 500 artists chosen from FreeDB, 37 were either repeated names (artists not identified by the approximate string match such as **Mozart** and **Mozart, Wolfgang Amadeus**), or else were not found in AllMusic, so our effective dataset had 463 examples. A number of artists that appeared to be more popular outside of North America were missing from AllMusic (eg. **Eri Esittajia**). Furthermore, of the more exotic artists that did appear in AllMusic, many entries were relatively incomplete—there was no background data on the artist, often nothing but a list of their albums and a genre (eg. **Die Artze**). This cast some doubt on the reliability of the genre for those artists. In contrast, popular artists (eg. **Radiohead**) had biographical sketches, a list of styles to add precision beyond genre, and a list of moods to which their music corresponds.

At the time of the experiments, AllMusic.com defined 32 popular and classical genres (though has risen to 35 at the time of writing). For the experiments below, the AllMusic-genre was represented as a probability distribution over the 32 AllMusic genres. For a given artist, probability mass was divided evenly among the genres to which his or her music belonged.

Algorithm	Accuracy		KL Loss	
	Mean	$\pm 95\%$	Mean	$\pm 95\%$
RANDOM	3.00%	0.07	3.665	0.0219
ROCK	47.09%	0.29	2.269	0.0374
LINSOFT	74.10%	0.20	0.922	0.0539

Table 5.1. Predictive power of FreeDB genre histograms.

5.3 Predictive Model

Since several of the genre labels in **FreeDB** are identical to genres that appear in **AllMusic**, I hoped that a relatively simple neural network would suffice to map FreeDB-genre to AllMusic-genre. I chose a map of the form given in equation 5.1 and Kullback-Leibler cost (defined in section 2.3.2). The neural network had no 'hidden layer'; it was a linear transform from the input space to a 32-dimensional real space, followed by a softmax transform that ensured the network's output was a valid distribution (all numbers positive, sum to one).

$$f_{fdb}(x) = \text{softmax}(Wx + b) \quad (5.1)$$

The parameters to optimize were the weights W and the output biases b . Optimization was done using the same delta-bar-delta batch-mode gradient descent method described in section 4.5, using early stopping for regularization. The early-stopping heuristic was to wait until 50 successive iterations failed to improve on validation performance.

5.4 Prediction Performance

The results of the experiment are posted in Table 5.1 and in Figure 5.2. In the table, three algorithms appear, and two scoring measures. The first algorithm RANDOM outputs a random histogram without making reference to the data in any way. The second algorithm, ROCK, guesses each output label according to the frequency in the training data, and represents the best classifier possible that does not use the input data. In this case, since *rock* is the most frequent label, I've dubbed this algorithm ROCK. The third algorithm LINSOFT is the neural network classifier described above, trained by gradient descent. The KL measure is the mean KL -divergence between test predictions and test targets (explained below in section 5.4.2). The 0 – 1 measure (classification accuracy) is the fraction of test examples such that the index of the maximum predicted value was equal to the index of an arbitrarily chosen maximum value in the target.

5.4.1 Classification Rates (1 – $L0$ – 1)

While this task is not strictly classification, $92\% = (435/463)$ of the artists in our dataset have a single genre, so the 0 – 1 measure is relevant. The expected performance of the RANDOM

algorithm is $\frac{1}{32} = 3\%$, in precise agreement with observation. The performance of ROCK is more interesting, it correctly labelled 47% of the data, which underscores the enormous class imbalance in **AllMusic**'s genre labels. The LINSOFT classifier correctly labelled 74% of test examples. This demonstrates a large degree of correlation between **FreeDB** and **AllMusic**.

Still, there are a large number of mistakes in view of the fact that the input and output are both genre histograms. One explanation is that a learning problem of these dimensions would normally be very difficult. Our model has over 17 000 parameters, and our dataset has only 463 examples, 338 of which are used to train, and 84 of which are used for validation. At the same time consider that each example is described by a histogram over 640 genres, and we would like to assign a histogram over 32 genres. Also consider that half of the target labels are *rock*, severely limiting the value of the little training data that we have. The fact that our model is able to learn anything at all is good evidence that the relationship between these two representations is simple, and that the early-stopping regularization method is effective.

One shortcoming of the 0 – 1 measure is that it does not take into account how close the model is to getting the answer right. Another way to compare model performance is by looking at the relative ranking of correct labels. As is seen in Figure 5.2 the LINSOFT classifier converges towards 100% classification more quickly than the other two, indicating that its highly-ranked choices are more often the right ones. For example, 89% of correct answers are found in the top 5 of 32 choices.

5.4.2 KL Divergence

The *KL* divergence between the target and predicted histograms provides another perspective on the performance of the algorithms. When there is just one correct genre for an artist, the *KL* divergence is the negative logarithm of the probability mass that the prediction put on the right genre. When there are multiple correct genres, the *KL* divergence is more difficult to interpret—it is the average difference (among correct labels) between the logarithm of the correct probability mass minus the logarithm of the predicted probability mass. The *KL* divergence is given in equation 5.2, in which P is the true target distribution and Q is the classifier's prediction.

$$KL(P||Q) = \sum_i P_i \log \left(\frac{P_i}{Q_i} \right) \quad (5.2)$$

Recalling that so many of the examples have just a single correct answer, a *KL* score of z means that the model got an average fraction of e^{-z} of the density that it was supposed to. In this way we can see that the *KL* score of RANDOM corresponds to probability mass 0.025, the score of ROCK corresponds to 0.10, while the score of LINSOFT corresponds to 0.40. With 32 genres in the target histogram, this represents a significant amount of learning.

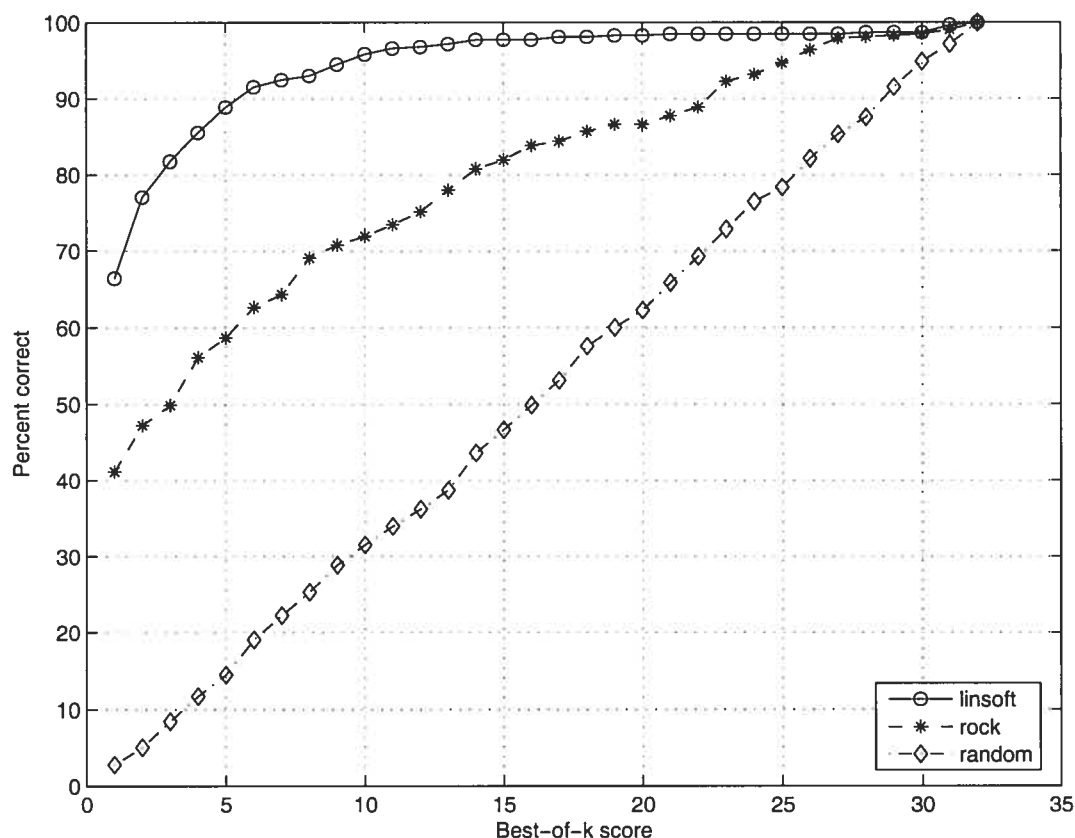


Figure 5.2. Cumulative sums showing the relative contributions in rank order. Faster convergence to 100% represents better performance in multi-task classification because it indicates that correct answers, even when not the first choice, are highly ranked by the classifier.

5.5 Discussion

The results suggest that a linear model explains much of the variation in **AllMusic** given **FreeDB** input, although not enough to say that **FreeDB** contains more information than **AllMusic**. Given that **AllMusic** is a gold-standard for stylistic music classification, **FreeDB** should be considered a useful resource for labelling music for the purpose of genre classification. One of the most interesting potential applications of this result is that large quantities of music can be automatically labelled with a reference genre. It remains to be seen how to compare a divergence with a **FreeDB**-genre with a notion of misclassification in the **AllMusic**-genre sense. Only after such a method of comparison is worked out could we consider using **FreeDB**-genre as a target instead of **AllMusic**-genre. It also remains to be seen how the method handles a larger set of artists and genres.

Chapter 6

Future Work and Conclusion

A short term goal would be to carry out the experiments described here with the ADABOOST implementation used in Bergstra et al. [2006a]. Some informal initial trials suggest that ADABOOST.MH (Schapire and Singer [1998]) is 4 or 5 percentage points better than NN_{hid} .

I've organized several general directions for future work below into medium-term and long-term projects.

6.1 Coarse Genre Classification

Following up on the audio-classification problem presented in chapter 4, it might be best to take a step back and recognize that for humans, genres and other patterns of similarity emerge only after listening to a large amount of unlabelled music. In the medium term, it might be interesting to apply existing algorithms of semi-supervised machine learning. The amount of accessible music tagged with genre pales in comparison to the amount of music that is not tagged. Future work could consider unsupervised and semi-supervised methods for learning genre boundaries in existing feature-spaces.

6.2 High-Resolution Genre Classification

Following up on the possibility of large-scale high-resolution genre learning problems raised in chapter 5, there are several directions for future work.

One obvious step from traditional genre-classification to this new ground would be to run traditional algorithms on the new problem. In the case of the algorithms presented in chapter 4, the adaptations would be minimal. On one hand, since there are so many genres in FreeDB (tags in LastFM) it may be harder to predict a distribution over them. On the other hand, these datasets are much larger so that it may be possible to learn complex class-wise distributions over existing feature spaces. LastFM, for example, refers to roughly 1000 times as much audio as Tzanetakis.

An important step along the way to taking advantage of FreeDB's and LastFM's histograms is to develop appropriate smoothing methods. Recall from figure 3.1 that many histograms in these datasets have very few points. Since it is the underlying distribution over genres/tags that we wish to predict, we must look to methods such as those developed for word n -gram models in natural-language processing or else invent alternatives (Manning and Schütze [1999]). Not only is it unclear how genre-histograms should be smoothed, but it is not even clear how to judge the fitness of a smoothing method outside a particular application.

A third project might look into the impact of the divergence that measures the difference between a prediction and a (smoothed) genre histogram. While KL divergence is the correct measure in certain applications, it may not be correct here. Given a certain model of how recommendations are made from predictions and of how listeners respond, it would be interesting to seek a divergence to maximize the efficiency of recommendation.

Later, we might pause to wonder if FreeDB-genre for artists (as I have defined it in 5) is any good as a recommendation guide. It is beyond the scope of a computer science project to conduct a large study, but instead we could return to AllMusic and run a second experiment. Recall that AllMusic describes artists with Style, Mood, and Similar Artists as well as Genre. FreeDB-genre would be vindicated as a genre if it could predict those attributes with good accuracy.

In the long term, we must recognize that while a stream of spectral phases and energies is a biologically plausible transformation of sound, none of the segmentation techniques presented here can yield the rich representation of audio that human listeners enjoy. For example, the segment method prevents a learning algorithm from characterizing note onsets, or any aspect of the way a particular singer's or instrument's sound evolve on any time scale. If we wish to make progress in the prediction of fine notions of genre for personalized recommendation and music navigation, we will need algorithms that can learn about instruments, singers, and families of rhythm. There are many principles and algorithms for learning problems involving sequences of feature vectors or even the raw samples themselves, and future work will consider their application to the domain of characterizing recorded audio.

6.3 Conclusion

In this work, I have made progress on the problem of how to apply machine learning and signal processing to classify music by genre. Though the technique of segmentation is not new, this work investigated the influence of segment number, length, and contiguity, found that a large number of overlapping, short (ideally around three seconds), and contiguous segments is consistently best or near-best in view of the uncertainty in performance estimates. No previous work has used these parameter choices, and they represent a significant improvement over popular approaches such as a small number of non-overlapping short segments, and a single whole-song segment. To my knowledge, the algorithm presented in Bergstra et al. [2006a] achieves the highest published rates on **Tzanetakis**, **USPOP.MIREX05** and **MAGNA.MIREX05**. That algorithm employed ADABOOST as a classifier and achieved an accuracy of 83% on **Tzanetakis**. When a simple three-

layer neural network replaced ADABOOST, the accuracy was 78%, but using a large number of overlapping, contiguous, short, segments and the mean and covariance in new new Mel-Scale Phon coefficients, performance rebounded to 81%. At the same time, the Mel-Scale Phon coefficients have just over half the dimensionality of the MIREX feature set, and are quicker to compute. The success of the Mel-Scale Phon coefficients also supports the point that the final Fourier transform in the computation of Mel-Frequency Cepstral Coefficients is not necessarily helpful when the goal is a high classification accuracy. All of these scores compare favourably with previous classification rates on this dataset, such as Li and Tzanetakis [2003] (71%), and Tzanetakis and Cook [2002] (61%).

Recent work by Lippens et al. [2004] and an earlier work by Soltau [1997] (described in Aucouturier and Pachet [2003]) suggest standards to which we may hold these results. Lippens et al. [2004] conducted a study in which people were asked to do genre-recognition on a small dataset involving six quite different genres. It was found that on average, people were correct only around 90% of the time. In a Turing-test conducted by Soltau [1997], he finds that his automatic music classifier is similar in performance to 37 subjects who he trained to distinguish *rock* from *pop* using the same training and test sets. Given the steady and significant improvement in classification performance since 1997, it is likely that automatic methods are already more efficient than some people at learning the sort of coarse genres in **Tzanetakis**.

I believe that the next step in the development of genre-classification algorithms is to predict finer notions of genre. To this end I have presented a method that associates an artist with a histogram over a large number of potential genres by using FreeDB records. I showed that these histograms account for a significant portion of genre labels in AMG's AllMusic database, so that we, as researchers in genre prediction, may see them as an inexpensive surrogate. This method is intended to facilitate the large-scale application of genre descriptors to music that is already labelled by artist for the purpose of training more powerful predictive models. Taking advantage of large datasets with these genre descriptors to learn more sophisticated models of music similarity is left for future work.

Bibliography

- P. Ahrendt, A. Meng, and J. Larsen. Decision time horizon for music genre classification using short time features. In *EUSIPCO*, pages 1293–1296, Vienna, Austria, Sep 2004.
- Peter Ahrendt and Anders Meng. Music genre classification using the multivariate ar feature integration model. Extended Abstract, 2005. MIREX genre classification contest (www.music-ir.org/evaluation/mirex-results).
- Alan V. Willsky Alan W. Oppenheim. *Signals and Systems*. Prentice Hall, 1996.
- J.J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):1–12, 2003.
- J.J. Aucouturier and F. Pachet. Music similarity measures: Whats the use?
- J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, , and M. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 2005.
- Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.
- J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 2006a. Accepted.
- J. Bergstra, A. Lacoste, and D. Eck. Predicting genre labels for artists using freedb. In *Proc. 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006b. Submitted.
- Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996.
- L. Breiman and P. Spector. Submodel selection and evaluation in regression: The x-random case. *International Statistical Review*, 60:291–319, 1992.
- Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- Rich Caruana. *Multitask Learning*. PhD thesis, Carnegie Mellon University School of Computer Science, 1997.

- T. Crawford and M. Sandler, editors. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, England, Sept 2005.
- R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2001.
- D. Eck and N. Casagrande. Finding meter in music using an autocorrelation phase matrix and shannon entropy. In *Proc. 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.
- Bradley Efron and Rob J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability Series*. CRC Press, 1994.
- Michael Fingerhut, editor. *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR 2002)*, Paris, France, 2002.
- B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, Berkeley, California., 2000.
- J.C Junqua and J.P Haton. *Robustness in Automatic Speech Recognition*. Kluwer Academic, Boston, 1996.
- B. Kedem. Spectral analysis and discrimination by zero-crossings. *Proc. IEEE*, 74(11):1477–1493, 1986.
- T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, , and A. Linney. Classification of audio signals using statistical features on time and wavelet tranform domains. In *Proc. Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP-98)*, volume 6, pages 3621–3624, 1998.
- Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.
- Tao Li and George Tzanetakis. Factors in automatic musical genre classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 2003.
- Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, New York, NY, USA, 2003. ACM Press.
- Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre recognition. In Crawford and Sandler [2005], pages 34–41.
- S. Lippens, J.P. Martens, and T. De Mulder. A comparison of human and automatic musical genre classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages iv–233– iv–236, 2004.
- Beth Logan. Mel frequency cepstral coefficients for music modeling. Plymouth, MA, USA., 2000.

- Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *2001 IEEE International Conference on Multimedia and Expo (ICME'01)*, page 190, 2001.
- J Makhoul. Linear prediction: A tutorial review. In *Proceedings of the IEEE*, volume 63, pages 561–580, 1975.
- Michael Mandel and Dan Ellis. Song-level features and SVMs for music classification. 2005a. MIREX 2005 genre classification contest (www.music-ir.org/evaluation/mirex-results).
- Michael I. Mandel and Daniel P.W. Ellis. Song-level features and support vector machines for music classification. In Crawford and Sandler [2005].
- Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, May 1999.
- C. McKay and I. Fujinaga. Automatic music classification and the importance of instrument identification. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM05)*, Montreal, Canada, 2005.
- A. Meng, P. Ahrendt, and J. Larsen. Improving music genre classification by short-time feature integration. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume V, pages 497–500, mar 2005.
- XuanLong Nguyen, Martin Wainwright, and Michael Jordan. Divergences, surrogate loss functions and experimental design. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1011–1018. MIT Press, Cambridge, MA, 2006.
- F. Pachet and D. Cazaly. A taxonomy of musical genres. In *Proc. Content-Based Multimedia Information Access (RIAO)*, 2000.
- Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classification. In Crawford and Sandler [2005].
- Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classification. In Crawford and Sandler [2005].
- D. Perrott and R. Gjerdingen. Scanning the dial: An exploration of factors in the identification of musical style. In *Society for Music Perception and Cognition Conference (SMPC)*, Evanston, IL, 1999.
- Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, November 2000.
- Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 80–91, New York, NY, USA, 1998. ACM Press. ISBN 1-58113-057-0.

Eric Scheirer and Malcolm Slaney. Construction and evaluation of a robust multifeature music/speech discriminator. Munich, Germany, 1997. IEEE.

Hagen Soltau. Erkennung von musikstilen. Master's thesis, Universitat Karlsruhe, 1997.

G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals.

George Tzanetakis, editor. *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006.

George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, Jul 2002.

Vladimir Vapnik. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons, 1998.

Kris West and Stephen Cox. Finding an optimal segmentation for audio genre classification. In Crawford and Sandler [2005].

Changsheng Xu, Namunu C.Maddage, Xi Shao, and Qi Tian. Musical genre classification using support vector machines. In *In International Conference of Acoustics, Speech & Signal Processing (ICASSP03)*, HongKong, China, 2003.

Appendix A

Software

I wrote a lot of software in the course of my Masters, and I plan to release at least the following two libraries on savannah.gnu.org. I think they are well-implemented, stable, and have programming interfaces that make them accessible and re-usable. The first is a library called `fextract` for extracting features from audio. The second is a library called `liblayerfb` that implements activation functions for neural networks.

A.1 `fextract`

`Fextract` is the library I used to extract acoustic features from an audio stream. It relies on efficient libraries to handle the linear algebra (BLAS), transcendental functions (`libacml_mv`), and Fourier transforms (FFTW3). Furthermore it was implemented to take advantage of the fact that many features share certain processing steps. For example, `rolloff`, `mfcc`, and `rceps` features all require a spectrogram, but then `rceps` and `mfcc` features require the log-magnitude of the spectrogram. `Fextract` is implemented as a sort of computational tree, in which any combination of features (leaves) can be calculated and the tree structure indicates which common calculations between multiple features need be calculated only once.

To give an idea of the performance of `fextract`, on an AMD 64 Processor (3200+) with a music database stored on a local hard disk, a feature-extraction of 128 RCEPS, 32 MFCC, 16 rolloff, zero-crossing rate, spectral centroid, spectral spread, and 32 LPC for each 47ms frame of the entire Tzanetakis database (30000 seconds) took 3 minutes, 7 seconds.

A.2 `liblayerfb`

`Liblayerfb` implements a variety of differentiable vector-valued functions. Most functions are offered in two flavours: one version accepts matrix arguments passed as `gsl_matrix` pointers, the other version accepts matrix arguments in the BLAS-style pairs. Each routine has a parameter list of the form (LAYER_OP, size arguments, auxiliary buffers, const value input buffers, value out-

put buffer, gradient-with-respect-to-input buffers, const gradient-with-respect-to-output buffer). Although argument lists can be complex when written out in full, the order was chosen to be easily readable when split over multiple lines.

Each routine can perform each of three tasks, indicated via the leading `LAYER_OP` parameter: validate the arguments, compute the value, and propagate the gradient. These tasks should be performed in order. Like `FFTW3`, these routines expect the same buffers to be supplied between calls. These routines don't return errors; Instead, they `assert()` the validity and soundness of the inputs where possible. If you haven't disabled `assert` statements, these routines will `abort()` at the point of detecting invalid input.

The priority throughout the implementation of this library was to minimize redundant calculation, to take advantage of cache re-use by using blocking algorithms for matrix operations, and to separate simple math operations from more complicated ones in order to make maximal use of a few platform-specific optimizations (e.g. a more efficient implementation of softmax for `x86_64` in assembly).

Appendix B

Last.FM Tag Sample

The following strings were taken randomly from a list of thirty thousand Last.fm tags.

Delete	beefy
layered	fuzz
Moosemans Jukebox	album closer
last track	FUCKING AWESOME
QaF	gracious intensity
my favourite tracks	dance like a fool
cmj new music monthly	Female fronted metal
symphonic metal	Yay
My Songs	lacuna coil nightwish evanescence
holiday	at work
lastfm radio discovery	rache65
hyper	Misspent Youth
Sheryl Crow	joy
n-se 2004	hell yah
optimistic rock	Stronger soundtrack
new music sampler	british trad rock
deprimiram se	Briternative
fast rock	best break-up songs
anti-war	Inglaterra
My Personal favorites	politic
sweet covers	oink
beefalo	good for driving around in a truck
angry muzak	rockn out
YAR	I Made It Through the Cub Scouts
favourite pink floyd songs	caffiene
Melancholy time	flange
favorite pink floyd songs	mmm mmm good
dream music	My Favorite Songs and Pieces
Kickin ass	Cursing

Appendix C

Preprint: Aggregate Features and AdaBoost for Music Classification

Aggregate features and ADABOOST for music classification

James Bergstra · Norman Casagrande ·
Dumitru Erhan · Douglas Eck · Balázs Kégl

Received: 11 October 2005 / Revised: 29 April 2006 / Accepted: 2 May 2006 / Published online: 30 June 2006
Springer Science + Business Media, LLC 2006

Abstract We present an algorithm that predicts musical genre and artist from an audio waveform. Our method uses the ensemble learner ADABOOST to select from a set of audio features that have been extracted from segmented audio and then aggregated. Our classifier proved to be the most effective method for genre classification at the recent MIREX 2005 international contests in music information extraction, and the second-best method for recognizing artists. This paper describes our method in detail, from feature extraction to song classification, and presents an evaluation of our method on three genre databases and two artist-recognition databases. Furthermore, we present evidence collected from a variety of popular features and classifiers that the technique of classifying features aggregated over segments of audio is better than classifying either entire songs or individual short-timescale features.

Keywords Genre classification · Artist recognition · Audio feature aggregation · Multiclass ADABOOST · MIREX

Editor: Gerhard Widmer

J. Bergstra (✉) · N. Casagrande · D. Erhan · D. Eck · B. Kégl
Department of Computer Science, University of Montreal, Montreal Quebec H3C 3J7, Canada

N. Casagrande

D. Erhan

D. Eck

B. Kégl

1. Introduction

Personal computers and portable digital music players are increasingly popular platforms for storing and playing music. (Apple alone has sold more than 42 million iPod players and more than a billion songs from their iTunes Store.) As personal music collections continue to grow, the task of organizing and selecting music becomes more challenging. This motivates research in automatic methods for search, classification, and recommendation.

We deal here with one aspect of this challenge, the prediction of the genre or artist of a song based on features extracted from its audio waveform. This task is challenging for at least two reasons. First, the raw musical signal is very high dimensional. In CD quality audio, a single monophonic waveform contains 44100 values per second. Thus a stereo 3 minute song contains nearly 16 million samples. More compact acoustic features can be extracted from the waveform. However, many of these are computed over short frames of audio, thus reducing the degree of compression achieved. Second, music databases can be very large. The website www.itunesregistry.com recently reported an average iTunes collection size of 3,542 songs. Commercial databases for online music services can contain 500,000 or more (personal communication, Benjamin Masse, president radiolibre.ca). To be useful for real world applications, a classifier must scale to large datasets.

In this paper we present a genre and artist classifier that addresses these issues. We use the ensemble learner ADABOOST, which performs large-margin classification by iteratively combining the weighted votes of a collection of weak learners. This approach has two advantages in the context of the challenges we face. First, our model uses simple decision stumps, each of which operates on a single feature dimension. Thus our model performs feature selection in parallel with classification. Although a feature set with redundant or non-informative features will slow down computation, those features will be left behind by ADABOOST as it iteratively selects informative features. Second, ADABOOST scales linearly with the number of training points provided. If we limit the number of learning iterations, our model has the potential to deal with very large datasets. This compares favorably with other state-of-the-art large margin classifiers such as SVMs, which have an overall computational complexity that is quadratic in the number of data points.

This paper makes two contributions. First we provide experimental evidence that our algorithm is competitive with other state-of-the-art classifiers for genre and artist recognition. Second we explore the issue of feature extraction, focusing on the challenge of aggregating frame-level acoustic features into a form suitable for classification. We analyze experiments on several classifiers using different aggregation segment sizes. The results support the general claim that aggregation is a useful strategy, and suggest reasonable limits on segment size.

The paper is structured as follows: in Section 2 we discuss previous attempts at music genre and artist recognition, focusing separately on feature extraction, feature aggregation and classification. In Section 3 we describe our ADABOOST model and aggregation technique. In Section 4 we provide experimental results for our model. This section includes a discussion of the results of the MIREX (Music Information Retrieval Evaluation eXchange) 2005 genre prediction contest and artist recognition contests where our algorithms won first and second prizes, respectively. In Section 5 we present the experimental results from our investigation of segment length for feature aggregation. Finally in Section 6 we offer concluding remarks and discuss future work.

2. Genre and artist classification

In the following section we treat genre and artist classification as a three-step process. While we do not provide a survey, we note that much previous work in music classification fits in this framework and describe it below. The first step is the extraction of acoustic features from short frames of audio data that capture information such as timbre and rhythm. The second step is the aggregation of frame-level features into more compressed segment-level features. The third step is the prediction of genre or artist for a song using these compressed features as input to a classifier.

2.1. Feature extraction and aggregation

To our knowledge there is no accepted theory of which features are best for music classification tasks such as genre and artist recognition. Many different methods have been tried, including Fourier analysis and related measures such as cepstral analysis (yielding MFCCs). In addition to the family of Fourier methods, results have been reported for wavelet analysis (Lambrou et al., 1998), autoregression (Ahrendt and Meng, 2005) and the collection of statistics such as zero-crossing rate, spectral centroid, spectral roll-off and spectral spread. A survey by Aucouturier and Pachet (2003) describes a number of popular features for music similarity and classification, and research continues (e.g. Bello et al. (2005), Pampalk et al. (2005)). In Section 3 we describe the specific features we use in our model.

In order to capture relatively fine-timescale structure such as the timbre of specific instruments, features are often extracted from short (~ 50 ms) frames of data. Our goal, however, is to predict something about an entire song which encompasses many frames (~ 20 per second in our implementation). This raises the question of how to use this frame level information.

One option is to compress these frame-level features into a *single* set of song-level features. Tzanetakis and Cook (2002) and Li et al. (2003) fit individual Gaussians to each feature (diagonal covariance among Gaussians). More recently, (Mandel and Ellis, 2005a) generated song-level features using Gaussian densities with full-covariance. Gaussian mixtures have also been used to generate song-level features by, e.g., Aucouturier and Pachet (2002) and Pampalk et al. (2005).

On the other end of the spectrum it is also possible to first classify directly the frame-level features themselves, and then to combine the individual classifiers into song labels by a majority vote. Xu et al. (2003), for example, reports good genre classification results on a small dataset using this approach. West and Cox (2004), however, note a significant improvement in performance when they use a “memory” feature, which is the mean and variance of frame-level features over the previous second.

A third option is to aggregate frame-level features over an audio segment that is longer than a single frame but shorter than an entire song. As in the second approach, individual segment classifications are combined to make a decision about an entire song. This technique was described in Tzanetakis and Cook (2002), who summarized a one-second ‘texture window’ with feature means and variances before performing classification. In a similar work, Ahrendt and Meng (2005) used an auto-regressive model in place of Gaussians. West and Cox (2005) used an onset detection algorithm to partition the song into segments that correspond to single notes.

While promising results have been reported for a wide range of segment sizes (all the way from the frame to the song), no one to our knowledge has undertaken a systematic exploration of the effects of segment size on classification error. In Section 5 we investigate this question by training several algorithms using the same dataset. For all experiments we

use the popular technique of fitting independent Gaussians (diagonal covariance) to each feature in a segment. We then vary segmentation size and analyze results for evidence that certain segment lengths work better than others.

2.2. Classification

A wide range of algorithms have been applied to music classification tasks. These include minimum distance and K-nearest neighbor in Lambrou et al. (1998), and Logan and Salomon (2001). Tzanetakis and Cook (2002) used Gaussian mixtures. West and Cox (2004) classify individual frames by Gaussian mixtures, Linear Discriminant Analysis (LDA), and regression trees. Ahrendt and Meng (2005) classify 1.2s segments using multiclass logistic regression.

Several classifiers have been built around Support Vector Machines (SVMs). Li et al. (2003) reported improved performance on the same dataset as Tzanetakis and Cook (2002) using both SVM and LDA. Mandel and Ellis (2005a) used an SVM with a kernel based on the symmetric KL divergence between songs. Their model performed particularly well at MIREX 2005, winning the Artist Recognition contest and performing well in the Genre Recognition contest as well. While SVMs are known to perform very well on small data sets, the quadratic training time makes it difficult to apply them on large music databases. This motivates research on applying equally well-performing but more time efficient algorithms to music classification.

3. Algorithm

In this section we describe a genre and artist classifier based on multiclass ADABOOST. In keeping with the organization of Section 2, we organize this section around feature extraction, feature aggregation, and classification.

3.1. Acoustic feature extraction and aggregation

For all experiments, we break an audio waveform into short frames (46.44ms, or 1024 samples of audio at 22050Hz). The feature sets for experiments presented in this work are drawn from the following list.¹ Those unfamiliar with standard methods of audio signal processing may refer to Kunt (1986) for background.

- *Fast Fourier transform coefficients (FFTCs)*. Fourier analysis is used to analyze the spectral characteristics of each frame of data. In general we computed a 512-point Fourier transform $\mathcal{F}(s)$ of each 1024-point frame s . The 32 lowest frequency points were retained for our experiments.
- *Real cepstral coefficients (RCEPS)*. Cepstral analysis is commonly used in speech recognition to separate vocal excitation from the effects of the vocal tract. See Gold and Morgan (2000) for an overview. RCEPS is defined as $\text{real}(\mathcal{F}'(\log(|\mathcal{F}(s)|)))$ where \mathcal{F} is the Fourier transform and \mathcal{F}' is the inverse Fourier transform.
- *Mel-frequency cepstral coefficients (MFCC)*. These features are similar to RCEPS, except that the input x is first projected according to the Mel-scale Junqua and Haton (1996), a psycho-acoustic frequency scale on which a change of 1 unit carries the same perceptual significance, regardless of the position on the scale.

¹ Our feature extractor is available as a C program from the first author's website: <http://www-etud.iro.umontreal.ca/~bergstrj>.

- *Zero-crossing rate (ZCR)*. The zero-crossing rate (ZCR) is the rate of sign-changes along the signal. In a signal with a single pitched instrument, the ZCR is correlated with the dominant frequency (Kedem, 1986).
- *Spectral spread, spectral centroid, spectral rolloff* Spectral spread and spectral centroid are measures of how power is distributed over frequency. Spectral spread is the variance of this distribution. The spectral centroid is the center of mass of this distribution and is thus positively correlated with brightness. Spectral rolloff feature is the α -quantile of the total energy in $|\mathcal{F}_s|$.
- *Autoregression coefficients (LPC)*. The k linear predictive coefficients (LPC) of a signal s are the product of an autoregressive compression of the spectral envelope of a signal. These coefficients can be computed efficiently from the signal's autocorrelation by the Levinson-Durbin recursion (Makhoul, 1975).

After computing these frame-level features, we group non-overlapping blocks of m consecutive frames into segments. We summarize each segment by fitting independent Gaussians to the features (ignoring covariance between different features). The resulting means and variances are the input to weak learners in ADABOOST.

3.2. Classification with ADABOOST

After extracting the segment-level features, we classified each segment independently using ADABOOST. The training set was created by labeling each segment according to the song it came from. ADABOOST Freund and Schapire (1997) is an *ensemble* (or *meta-learning*) method that constructs a classifier in an iterative fashion. It was originally designed for binary classification, and it was later extended to multi-class classification using several different strategies. In this application we decided to use ADABOOST.MH Schapire and Singer (1998) due to its simplicity and flexibility.

In each iteration t , the algorithm calls a simple learning algorithm (the *weak learner*) that returns a classifier $\mathbf{h}^{(t)}$ and computes its coefficient $\alpha^{(t)}$. The input of the weak classifier is a d -dimensional observation vector $\mathbf{x} \in \mathbb{R}^d$ containing the features described in Section 2.1, and the output of $\mathbf{h}^{(t)}$ is binary vector $\mathbf{y} \in \{-1, 1\}^k$ over the k classes. If $h_\ell^{(t)} = 1$ the weak classifier “votes for” class ℓ whereas $h_\ell^{(t)} = -1$ means that it “votes against” class ℓ . After T iterations, the algorithm outputs a vector-valued discriminant function

$$\mathbf{g}(\mathbf{x}) = \sum_{t=1}^T \alpha^{(t)} \mathbf{h}^{(t)}(\mathbf{x}).$$

To obtain a single label, we take the class that receives the “most vote”, that is, $f(\mathbf{x}) = \arg \max_{\ell} g_{\ell}(\mathbf{x})$.

When no a-priori knowledge is available for the problem domain, small decision trees or, in the extreme case, *decision stumps* (decision trees with two leaves) are often used as weak classifiers. Assuming that the feature vector values are ordered beforehand, the cost of the weak learning is $O(nkd)$, so the whole algorithm runs in $O(nd(kT + \log n))$ time. In the rest of the paper, we will refer to this version of ADABOOST.MH as AB.STUMP. We also experimented with small decision trees as weak learners, and we refer to this version of ADABOOST.MH as AB.TREE.²

² Our implementation of ADABOOST.MH is available as a C++ program from <http://sourceforge.net/projects/multiboost>.

Table 1 Summary of databases used in our experiments

Database Type	Magnatune		USPOP		Tzanetakis
	Genre	Artist	Genre	Artist	Genre
Number of training files	1005	1158	940	1158	800
Number of test files	510	642	474	653	200
Number of classes	10	77	6	77	10
Average song length		~ 200s			30s

Note that if each weak classifier depends on only one feature (as a decision stump does) and the number of iterations T is much less than the number of features d , then ADABOOST.MH acts as an implicit feature extractor that selects the T most relevant features to the classification problem. Even if $T > d$, one can use the coefficients $\alpha^{(t)}$ to order the features by their relevance.

4. Experiment A—MIREX 2005

In this section we present results from our entry in the 2005 MIREX (Music Information Retrieval Evaluation eXchange) contest. MIREX is an annual series of contests whose main goal is to present and compare state-of-the-art algorithms from the music information retrieval community. It is organized in parallel with the ISMIR conference. We participated in two contests: Audio Genre Classification Bergstra et al. (2005a), and Audio Artist Identification Bergstra et al. (2005b). Both contests were straightforward classification tasks. Two datasets were used to evaluate the submissions: Magnatune³, and USPOP⁴. Both databases contain mp3 files of commercially produced full-length songs. The Magnatune database has a hierarchical genre taxonomy with 10 classes at the most detailed level (ambient, blues, classical, electronic, ethnic, folk, jazz, new age, punk, rock), whereas the USPOP database has 6 genres (country, electronic and dance, new age, rap and hip hop, reggae, rock). Each of the datasets has a total of 77 artists. Table 1 summarizes the parameters of the experimental setup.

Competition rules demanded that every algorithm must train a model and make a class prediction for the elements of the test set within 24 hours. Due to the large number of contest submissions, contest organizers could not perform a K -fold cross validation of results. Only a single train/test run was performed for each <entry, database> pair.

4.1. Parameter tuning

The 24 hour limit on the training time required us to take extra care in setting the hyper-parameters of the algorithm: the number of boosting iterations T and the number of frames per segment m . Since the contest databases Magnatune and USPOP were kept secret, we tuned our algorithm using a database of song excerpts, furnished to us by George Tzanetakis. This is the same dataset used in Tzanetakis et al. (2002), Tzanetakis and Cook (2002), and Li and Tzanetakis (2003). This database has a total of 1000 30-second song openings. Each excerpt is labeled as one of ten genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock). To our ears, the examples are well-labeled, and exhibit a wide range of styles and instrumentation within each genre. Although the artist names are not associated

³ www.magnatune.com

⁴ www.ee.columbia.edu/~dpwe/research/musicsim/uspop2002.html

with the songs, our impression from listening to the music is that no artist appears twice. Thus, the so-called *producer effect*, observed in Pampalk et al. (2005) is not a concern for these parameter tuning experiments.⁵

To tune the hyper-parameters we extracted frame-level features (256 RCEPS, 64 MFCC, 32 LPC, the lowest 32 of 512 Fourier coefficients, 16 spectral rolloff, 1 LPC error, and the zero-crossing rate) from the Tzanetakis database, and estimated parameter performance by 5-fold cross-validation. This gave us 800 training points, slightly lower than in the MIREX contests. More importantly, the length of a song excerpt in the Tzanetakis database is 30s, much shorter than the average length of a commercially produced song which we estimated to be ~ 200 s. This means that using the same segment length m and number of iterations T , 24 hours of training on the Magnatune and USPOP databases would be roughly equivalent to 3 hours of training on the Tzanetakis database. Thus, we set the time limit in our tuning experiments to 2 hours.

Although ADABOOST is relatively resistant to overfitting, the test error may increase after a large number of iterations. To avoid overfitting, we can validate T using data held-out from the training set. In our trials, we did not observe any overfitting so we decided not to validate T . Instead we let the algorithm run the longest possible within the time limit. In fact, given the time limit, underfitting was a more important concern. Since the number of training examples n (and so the training time) is inversely proportional to the segment length m , selecting a small m might prevent our algorithm from converging within 24 hours. In Section 5 we present an experimental study in which we determined that the optimal segment length is between 50 and 100 frames, if ADABOOST is permitted to converge. Based on our performance on the Tzanetakis database, however, we estimated that setting $m = 100$ could result in our algorithm being terminated before converging. Thus, to avoid underfitting, in the contest submission we set $m = 300$ which corresponds to segments of length 13.9s. Since the Tzanetakis database did not contain different song excerpts from the same authors, we did not tune the algorithm separately for the artist recognition contest: our submissions in the two contests were identical.

4.2. Results

The overall performance of each entry was calculated by averaging the raw classification accuracy on USPOP with a hierarchical classification accuracy on Magnatune.⁶ Table 2 and Table 3 summarize the contest results.⁷

Our submissions based on AB.STUMP and AB.TREE ranked first and second on both genre tasks. On the artist databases, our algorithms placed first and third, and second and third.

Note that all the tests were done on single folds (not cross-validation) and there were only around 500 test files in each database, so the significance of the differences is difficult to assess. However we can assert that in trials before the contest, our model obtained a classification rate of 83% on the Tzanetakis database. This compares favorably with the best published classification rate on the Tzanetakis database, of 71% obtained by Li et al. (2003).

⁵ The producer effect comes from the fact that songs from the same album tend to look the same through the lens of popular feature-extractors. If an album is split between training and testing sets, then we can expect artificially high test-set performance.

⁶ More details regarding the evaluation procedure can be found at http://www.music-ir.org/mirex2005/index.php/{Audio_Genre_Classification,Audio_Artist_Identification}.

⁷ More detailed results, including class confusion matrices and brief descriptions of each algorithm, can be found at <http://www.music-ir.org/evaluation/mirex-results/audio-{genre,artist}/index.html>.

Table 2 Summarized results for the genre recognition contest at MIREX 2005

Rank	Participant	Overall	Magnatune	USPOP
1	AB.TREE	82.34%	75.10%	86.92%
2	AB.STUMP	81.77%	74.71%	86.29%
3	Mandel & Ellis	78.81%	67.65%	85.65%
4	West, K.	75.29%	68.43%	78.90%
5	Lidy & Rauber [1]	75.27%	67.65%	79.75%
6	Pampalk, E.	75.14%	66.47%	80.38%
7	Lidy & Rauber [2]	74.78%	67.65%	78.48%
8	Lidy & Rauber [3]	74.58%	67.25%	78.27%
9	Scaringella, N.	73.11%	66.14%	75.74%
10	Ahrendt, P.	71.55%	60.98%	78.48%
11	Burred, J.	62.63%	54.12%	66.03%
12	Soares, V.	60.98%	49.41%	66.67%
13	Tzanetakis, G.	60.72%	55.49%	63.29%

Table 3 Summarized results for the artist identification contest at MIREX 2005

Rank	Participant	Mean performance	Magnatune	USPOP
1	Mandel & Ellis	72.45%	76.60%	68.30%
2	AB.STUMP	68.57%	77.26%	59.88%
3	AB.TREE	66.71%	74.45%	58.96%
4	Pampalk, E.	61.28%	66.36%	56.20%
5	West & Lamere	47.24%	53.43%	41.04%
6	Tzanetakis, G.	42.05%	55.45%	28.64%
7	Logan, B.	25.95%	37.07%	14.83%

5. Experiment B—Choosing segment length

In this section we investigate the effect of segment length on classification error. Selecting the correct segment length m is crucial both for achieving the highest possible classification accuracy and for calibrating training time. In the experiments described below we examine how the segment length affects classification accuracy given unlimited training time. Based on our experiments we conclude that for several popular genre prediction algorithms, the optimal segment length is around 3.5 seconds, with good values ranging from approximately 2 to 5 seconds.

To pursue this question, we tested four classifiers on the Tzanetakis database: AB.STUMP, AB.TREE, sigmoidal neural network (ANN), and SVM with Gaussian kernel. The ANN Bishop (1995) was a feed-forward network of 64 hidden nodes, fit by gradient descent. An implicit L_2 -norm weight-decay was introduced by initializing the network parameters to small values, and performing early-stopping using a validation set. No explicit regularization was applied during optimization. Classification was performed by a bagged ensemble of 10 networks, as described in Breiman (1996). To train our SVM Cortes and Vapnik (1995), we optimized the width of the kernel first—using a relatively low value of the soft-margin parameter—and then optimized the soft-margin parameter C , while fixing the optimal width. Both of the optimizations were done on a validation set and the optimal set of hyperparameters was selected by 5-fold cross-validation. Both versions of ADABOOST.MH were trained for 2500 iterations, which sufficed to reach a plateau in performance.

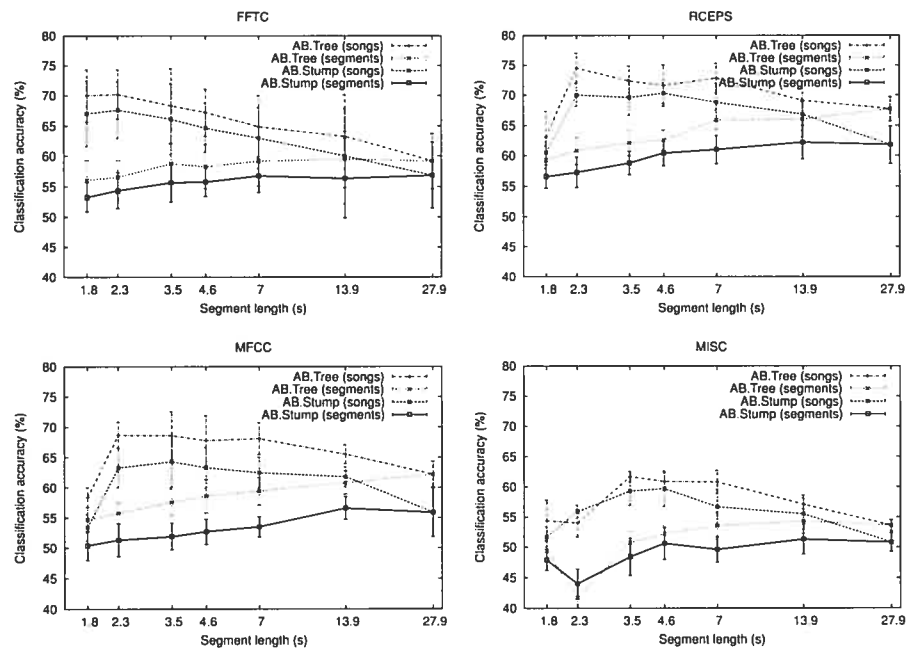


Fig. 1 The classification accuracy on four different feature sets trained with ADABOOST.MH using segment lengths $m \in \{40, 50, 75, 100, 150, 300, 600\}$

To evaluate the robustness of the segmentation procedure across different feature types, we ran separate experiments with four different feature sets described in Section 2: 64 MFCC, 128 RCEPS, 128 FFTC, and 19 MISC (including 1 ZCR, 1 Spectral Centroid, 1 Spectral Spread, and 16 Rolloff). Each of the 16 <feature set, classifier> pairs was evaluated using $m \in \{40, 50, 75, 100, 150, 300, 600\}$ frames/segment (corresponding to 1.8s, 2.3s, 3.5s, 4.6s, 7.0s, 13.9s, and 27.9s, respectively). All tests were done using 5-fold cross-validation. For all tests, we split the dataset at the song level to ensure that the segments from a single song were put into either the training or test set.

Figures 1 and 2 show the results. In all experiments, the general trend was that the segment classification rate rose monotonically with the segment length, and the whole-song classification rose and then fell. Although the fold variance makes it difficult to identify a clear peak, it appears that whole-song performance is optimal at around 3.5s segments across feature types and classification algorithms, except for the SVM that seems to prefer slightly longer segments. In the case of the neural network, we do not see the same degradation in segment-level performance as with ADABOOST, but we see the same rise and fall of the whole-song performance. The performance of the SVM is comparable to that of ADABOOST and the neural net when we use MFCC and RCEPS features, but it is much worse on the FFTC and MISC features. Note that it was impractical to obtain results using the 2.3- and 1.8-second segments with the SVM due to the training time which is quadratic in the number of training points. Figure 3 summarizes the results for every feature and algorithm on an entire song and on a segment size of 3.5 seconds.

The general rise and fall of the song-level classification can be explained in terms of two conflicting effects. On one hand, by partitioning songs in to more pieces, we enlarge our training set. This is universally good for statistical learning, and indeed, we see the benefits in

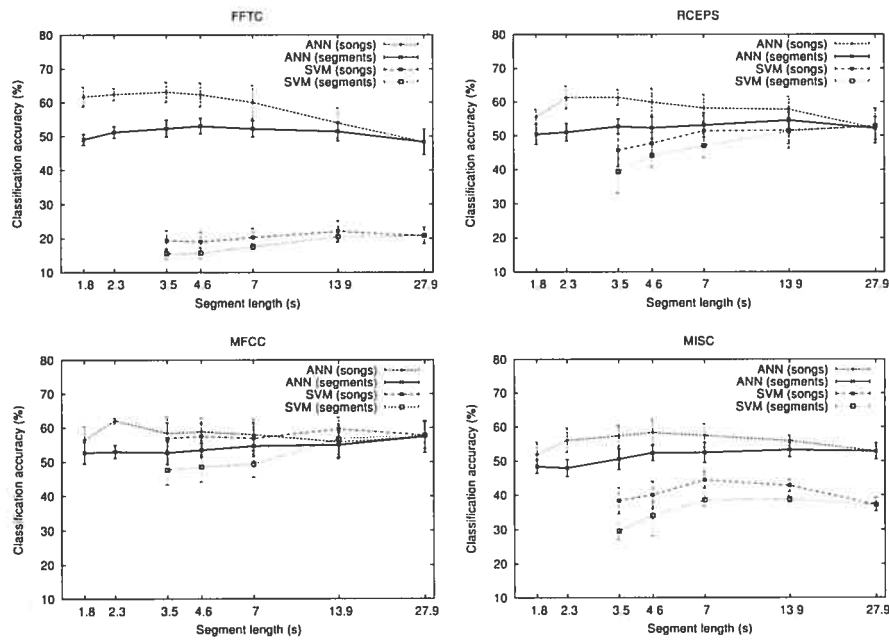


Fig. 2 The classification accuracy on four different feature sets trained with ANN and SVM using segment lengths $m \in \{40, 50, 75, 100, 150, 300, 600\}$

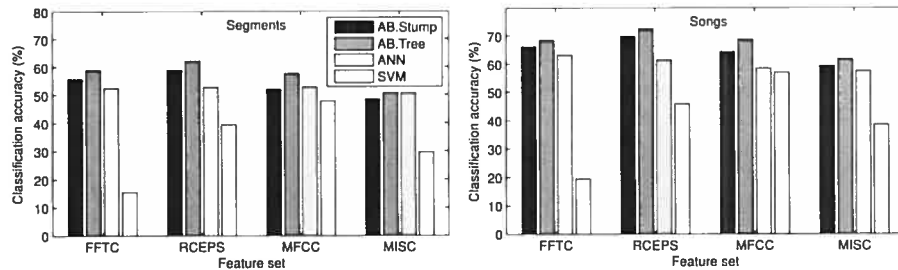


Fig. 3 The classification accuracy on four feature types and four classification algorithms using 3.5s segments

each classifier. On the other hand, smaller segments mean that our aggregate features (frame-level sample-means and sample-variances) have higher variance. For small-enough segments, the value of aggregating frame-level features is null, and previous work, particularly West and Cox (2004), supports the general claim that some amount of aggregation is necessary for good performance.

6. Conclusions and future work

In this work, we have presented a supervised learning algorithm for classifying recorded music. We have demonstrated with results on three databases that this algorithm is effective in genre prediction; to our knowledge, at the time of writing these results are the highest genre prediction rates published for each of Tzanetakis' database, Magnatune, and for the

USPOP database. We attribute this effectiveness to our particular strategy for aggregating frame-level, and to the use of ADABOOST to produce a classifier.

Recent work by Lippens et al. (2004) and an earlier work by Soltau (1997) (described in Aucouturier and Pachet (2003)) suggest standards to which we may hold these results. Lippens et al. (2004) conducted a study in which people were asked to do genre-recognition on a small dataset involving six quite different genres. It was found that on average, people were correct only around 90% of the time. In a Turing-test conducted by Soltau (1997), he finds that his automatic music classifier is similar in performance to 37 subjects who he trained to distinguish rock from pop using the same training and test sets. Given the steady and significant improvement in classification performance since 1997, we wonder if automatic methods are not already more efficient at learning genres than some people.

Future work would explore the value of segmentation when additional frame-level features act on longer frames to compute rhythmic fingerprints and short-term timbre dynamics. This includes evaluating features extracted from the beat histogram Tzanetakis and Cook (2002) and the autocorrelation phase matrix Eck and Casagrande (2005). We could also enrich the label set. Although in our experiments we considered a single label for each observation (each song belongs to one and only one category), it is more realistic to use hierarchical, or generally overlapping class labels for labeling music by its style. ADABOOST.MH extends naturally to *multi-label* classification by allowing the label vectors to contain more than one positive entry.

Music classification algorithms are approaching a level of maturity where they can aid in the hand labeling of data, and can verify large labeled collections automatically. However, to be useful in a commercial setting, these algorithms must run quickly and be able to learn from training sets that are one, two, even three orders of magnitude larger than the ones dealt with in our experiments. To this end, it would be useful to establish large databases of music that can be legally shared among researchers.

Acknowledgements Thanks to Alexandre Lacoste for many interesting ideas in conversation. Thanks also to George Tzanetakis for providing the genre classification dataset. This research is supported by grants from the Quebec Fund for Research in Nature and Technology (FQRNT) as and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- Ahrendt, P., & Meng, A. (2005). Music Genre Classification using the multivariate AR feature integration model. Extended Abstract. MIREX genre classification contest (www.music-ir.org/evaluation/mirex-results).
- Aucouturier, J., & Pachet, F. (2002). Music Similarity Measures: Whats the Use?. In: Fingerhut, M. (ed.): *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR 2000)*.
- Aucouturier, J., & Pachet, F. (2003). Representing musical genre: A state of the art. *Journal of New Music Research* 32(1), 1–12.
- Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. (2005). A Tutorial on Onset Detection in Music Signals. *IEEE Transactions on Speech and Audio Processing*.
- Bergstra, J., Casagrande, N., & Eck, D. (2005a). Artist Recognition: A Timbre- and Rhythm-Based Multiresolution Approach. MIREX artist recognition contest.
- Bergstra, J., Casagrande, N., & Eck, D. (2005b). Genre Classification: A Timbre- and Rhythm-Based Multiresolution Approach. MIREX genre classification contest.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.
- Breiman, L. (1996). Bagging Predictors. *Machine Learning* 24(2), 123–140.
- Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning* 20(3), 273–297.

- Crawford, T., & Sandler, M. (eds.) (2005). Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005).
- Eck, D., & Casagrande, N. (2005). Finding Meter in Music Using an Autocorrelation Phase Matrix and Shannon Entropy. In: *Proc. 6th International Conference on Music Information Retrieval (ISMIR 2005)*.
- Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139.
- Gold, B., & Morgan, N. (2000). *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley.
- Junqua, J., & Haton, J. (1996). *Robustness in Automatic Speech Recognition*. Boston: Kluwer Academic.
- Kedem, B. (1986). Spectral analysis and discrimination by zero-crossings. *Proc. IEEE* 74(11), 1477–1493.
- Kunt, M. (1986). *Digital Signal Processing*. Artech House.
- Lambrou, T., Kudumakis, P., Speller, R., Sandler, M., & Linney, A. (1998). Classification of audio signals using statistical features on time and wavelet transform domains. In: *Proc. Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP-98)*, 6, 3621–3624.
- Lippens, S., Martens, J., & De Mulder, T. (2004). A comparison of human and automatic musical genre classification. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4, 233–236.
- Li, T., Ogiwara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. In: *SIGIR 03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA, (pp. 282–289) ACM Press.
- Li, T., & Tzanetakis, G. (2003). Factors in automatic musical genre classification. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.
- Logan, B., & Salomon, A. (2001). A music similarity function based on signal analysis. In: *2001 IEEE International Conference on Multimedia and Expo (ICME'01)*. (p. 190).
- Makhoul, J. (1975). Linear Prediction: A Tutorial Review. In: *Proceedings of the IEEE*, 63, 561–580.
- Mandel, M. I., & Ellis, D. P. (2005a). Song-level features and support vector machines for music classification. In Crawford and Sandler (2005).
- Mandel, M., & Ellis, D. (2005b). Song-level features and SVMs for music classification. Extended Abstract. MIREX 2005 genre classification contest (www.music-ir.org/evaluation/mirex-results).
- Pampalk, E., Flexer, A., & Widmer, G. (2005). Improvements Of Audio-Based Music Similarity And Genre Classification. In (Crawford and Sandler, 2005).
- Schapire, R. E., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In: *COLT 98: Proceedings of the eleventh annual conference on Computational learning theory*. New York, NY, USA, (pp. 80–91) ACM Press.
- Soltau, H. (1997). Erkennung von Musikstilen. Masters thesis, Universitat Karlsruhe.
- Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10(5), 293–302.
- Tzanetakis, G., Ermolinskyi, A., & Cook, P. (2002). Pitch histograms in audio and symbolic music information retrieval. In: Fingerhut, M. (ed.): *Proceedings of the Third International Conference on Music Information Retrieval: ISMIR 2002*, 31–38.
- West, K., & Cox, S. (2004). Features and classifiers for the automatic classification of musical audio signals. In: *Proc. 5th International Conference on Music Information Retrieval (ISMIR 2004)*.
- West, K., & Cox, S. (2005). Finding an Optimal Segmentation for Audio Genre Classification. In (Crawford and Sandler, 2005).
- Xu, C., Maddage, N.C., Shao, X., & Tian, Q. (2003). Musical Genre Classification Using Support Vector Machines. In: *International Conference of Acoustics, Speech & Signal Processing (ICASSP03)*.

